

# An Introduction to Lifelong Supervised Learning

---

**Shagun Sodhani**  
FAIR, Meta AI

**Mojtaba Faramarzi**  
Universite de Montreal

**Sanket Vaibhav Mehta**  
Carnegie Mellon University

**Pranshu Malviya**  
Polytechnique Montréal

**Mohamed Abdelsalam**  
Universite de Montreal

**Janarthanan Rajendran**  
Universite de Montreal

**Sarath Chandar**  
Polytechnique Montréal  
Canada CIFAR AI Chair  
Quebec Artificial Intelligence Institute (Mila)

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Artificial Intelligence Systems . . . . .	1
1.2	Success Stories of Machine Learning . . . . .	4
1.3	Lifelong Learning Systems . . . . .	5
1.4	Outline . . . . .	7
1.5	Scope . . . . .	8
1.6	Target Audience . . . . .	9
<b>2</b>	<b>Overview of Lifelong Learning</b>	<b>10</b>
2.1	What is Lifelong Learning . . . . .	10
2.2	Background: Supervised Learning . . . . .	11
2.3	Lifelong Learning Formulation . . . . .	12
2.4	Prominent Scenarios in Lifelong Learning . . . . .	13
2.5	An overview of Lifelong Learning strategies . . . . .	14
2.6	Desiderata of Lifelong Learning Systems . . . . .	17
2.7	Relation to Other Areas . . . . .	18
2.8	Common Metrics in Lifelong Learning . . . . .	23
<b>3</b>	<b>Regularization-based Approaches</b>	<b>30</b>
3.1	Definition . . . . .	31
3.2	Importance-Based Regularization . . . . .	32
3.3	Bayesian-Based Regularization . . . . .	42

3.4	Distillation-based Regularization . . . . .	48
3.5	Optimization Trajectory based Regularization . . . . .	50
3.6	Summary . . . . .	56
<b>4</b>	<b>Memory-based Approaches</b>	<b>58</b>
4.1	A Unified View of Episodic Memory for Lifelong Learning . . . . .	60
4.2	Test-time use of Episodic Memory . . . . .	66
4.3	Memory Read & Write Sampling Strategies . . . . .	69
4.4	Generative Replay . . . . .	75
4.5	Summary . . . . .	77
<b>5</b>	<b>Architecture-based Approaches</b>	<b>79</b>
5.1	Modular Networks . . . . .	80
5.2	Parameter Isolation Systems . . . . .	91
5.3	Summary . . . . .	104
<b>6</b>	<b>Benchmarks</b>	<b>106</b>
6.1	Vision Benchmarks . . . . .	106
6.2	NLP Benchmarks . . . . .	119
6.3	Summary . . . . .	124
<b>7</b>	<b>Future Challenges</b>	<b>126</b>
	<b>References</b>	<b>129</b>

# 1

---

## Introduction

---

### 1.1 Artificial Intelligence Systems

**Artificial Intelligence** Artificial Intelligence (AI) systems can be defined as systems that think and act rationally like humans (Bellman, 1978; Kurzweil *et al.*, 1990; Schalkoff, 1991; Rich and Knight, 1992; Winston, 1992; Haugeland, 1997; Russell and Norvig, 2005). While the term was formally coined at the famous Dartmouth conference in 1956 (McCarthy *et al.*, 2006; Woo, 2014), philosophers dating back to Aristotle and Plato contemplated formulating the law governing the rational part of the mind. The idea of creating *intelligent* systems inspired myths like the story of Talos, a giant bronze robot created by gods that carried within it a mysterious life source and guarded the island of Crete (Shashkevich, 2019). Since then, psychologists, behaviorists, cognitive scientists, linguists, and computer scientists have championed various approaches for understanding intelligence and developing AI systems.

Early AI systems were often *rule-based*: given a collection of *rules of the world*, they would use approaches like search and symbol manipulation to solve a given task. These systems focused on (and generally performed well) on reasoning-related problems, like proving theorems (e.g.,

Logic Theorist (Gugerty, 2006) and General Problem Solver (Newell and Shaw, 1959)) or focused on setups with few entities to interact with (Minsky and Papert, 1972). Systems relying on these classical approaches enabled significant breakthroughs like IBM’s Deep Blue system defeating the then world champion of Chess in 1997. However, these systems were often limited by how fast they could *process* the rules. As a result, these systems do not work well when the number of combinations of rules becomes large. Another significant limitation of the rule-based systems is that they need a clean and well-curated collection of rules to start with. It is possible that one can define and describe these rules explicitly for a game like Chess, but this is often not feasible in real-life scenarios.

**Machine Learning** The over-reliance of early AI systems on hard-coded knowledge limited their scope and use for complex setups and real-world applications. Machine Learning (ML) is a sub-field of AI that aims to address this limitation by inferring knowledge from raw data using techniques like pattern mining, association rule mining, representation learning, classification, regression, etc. Machine Learning systems can be broadly categorized into two groups:

1. **Parametric models** are models that “summarize” (or encode) the knowledge in the given dataset/task<sup>1</sup> using a set of parameters. These models generally assume that a function exists that explains the knowledge in the data and infer the parameters of that function. Once the parameters have been learned, the original data is no longer needed. Common examples of parametric models include logistic regression, linear discriminant analysis, and neural networks.
2. **Non-parametric models** do not infer any parameters from the given data, though they may infer some summary statistics, like mean, to speed up inference. Common examples of parametric models include k-Nearest Neighbors and Support Vector Machines.

---

<sup>1</sup>For simplicity, we use the terms dataset and task interchangeably in the introduction

In general, a machine learning system may or may not have to *learn* feature representations for a given dataset. For example, consider an email spam classifier where the input to the system is a set of features like “is the email from an unknown user” or if certain keywords are present or not. In this case, these features could be fed as input to a logistic regression classifier, and only the classifier needs to be trained. In general, we can not assume access to high-quality, informative features, and the machine learning system has to infer these features. For example, in the email spam classifier example, the system may only have access to the blob of email text. It would need to learn a good feature representation that can be used as input to the classifier. In this case, the system could use a non-parametric approach like *term frequency-inverse document frequency* (TF-IDF) (Ramos *et al.*, 2003) or use a parametric representation learning model like a recurrent neural network (Hochreiter and Schmidhuber, 1997b; Cho *et al.*, 2014)

**Deep Learning** Deep Learning is a sub-field within machine learning that focuses on representation learning (learning representation from the given data), usually using parametric models. The high-level idea behind deep learning is as follows: There are some base computational units called layers, like the convolutional neural network layer (LeCun *et al.*, 1989), which can be stacked over each other (or, in general, composed arbitrarily) to create powerful architectures. For example, the ResNet architecture is composed using a stack of convolutional layers, along with other layers like max-pooling layers.

As the feature representation passes through the subsequent layers, it is transformed into more complex features. The resulting feature could be used as input to a classifier system. The entire system, i.e., the representation learning system, and the classifier system, can be trained together end-to-end. Today, machine learning is one of the most popular AI paradigms, and deep learning is the most popular representation learning approach. It is worth noting that the current AI systems are often a combination of techniques from different sub-fields. For example, AlphaGo (Silver *et al.*, 2016), which defeated the world champion of Go, uses convolution networks, a deep learning approach, to learn feature representation, and Monte-Carlo Tree Search, a traditional AI approach,

to search for the next action.

## 1.2 Success Stories of Machine Learning

Machine Learning Systems have come a long way since the McCulloch-Pitts Neuron, the first computational model of a neuron (McCulloch and Pitts, 1943). ML systems have shown impressive results in a number of problem settings where the previous AI approaches struggled: fundamental sciences (Gemp *et al.*, 2021; Pfau *et al.*, 2020; Bapst *et al.*, 2020), bio-medicine (Cireřan *et al.*, 2013; Litjens *et al.*, 2016), life-sciences (Senior *et al.*, 2020; Yim *et al.*, 2020; Tomařev *et al.*, 2019; Leibo *et al.*, 2018), hardware design and manufacturing (Schmidt *et al.*, 2019; Bhuvaneswari *et al.*, 2021; Mirhoseini *et al.*, 2021), graph analysis (Tang *et al.*, 2015; Kipf and Welling, 2016; Hamilton *et al.*, 2017), neuroscience (Mathis *et al.*, 2018; Mathis and Mathis, 2020) etc.

Even for domains where traditional AI systems were used earlier, the current generation of ML systems have led to significant improvements. This includes areas such as image understanding (Krizhevsky *et al.*, 2012; Xie *et al.*, 2017), semantic segmentation (Girshick *et al.*, 2014; Ren *et al.*, 2015), video processing (Fan *et al.*, 2021), machine translation (Bahdanau *et al.*, 2014; Cho *et al.*, 2014), question answering (Lan *et al.*, 2019; Zhang *et al.*, 2020d), text summarization (Raffel *et al.*, 2019; Lewis *et al.*, 2019), text generation (Radford *et al.*, 2019; Kaplan *et al.*, 2020), speech recognition (Schneider *et al.*, 2019; Baevski *et al.*, 2020), textless NLP (Lakhotia *et al.*, 2021; Kharitonov *et al.*, 2021; Polyak *et al.*, 2021),

robotics (Hadsell *et al.*, 2008; Koutnik *et al.*, 2013; Chen *et al.*, 2015a), social network analysis (Sodhani *et al.*, 2019; Tang and Matteson, 2021),

etc. ML systems have reached super-human performance on several tasks (Hochreiter and Schmidhuber, 1997b; Bahdanau *et al.*, 2014; Graves *et al.*, 2014; Mnih *et al.*, 2015; He *et al.*, 2016; Miller *et al.*, 2016; Vaswani *et al.*, 2017; Krizhevsky *et al.*, 2017; Silver *et al.*, 2017; Silver *et al.*, 2018; Devlin *et al.*, 2018; Vinyals *et al.*, 2019; Zhang *et al.*, 2020c; Brown *et al.*, 2020; Schrittwieser *et al.*, 2020; Badia *et al.*, 2020). These ML systems were already used in the digital world (Lewis-

Kraus, 2016; Zhai *et al.*, 2017; Naumov *et al.*, 2019) but are now being actively deployed in the physical world as well (Satariano and Metz, 2020; Vincent, 2021; Alex Davies, 2021).

These advances are bringing the current generation of AI systems closer to the long-standing goal of AI practitioners - designing systems that can *imitate* the behavior of humans or can demonstrate human-like general intelligence (TURING, 1950). However, despite all the success and promising results, there are still significant gaps in the capabilities of even the most powerful AI systems when compared to humans.

### 1.3 Lifelong Learning Systems

A key criticism of the current machine learning systems is that they tend to be *data-hungry* (Marcus, 2018; Ford, 2018). Take the example of the *GPT-3* model (Brown *et al.*, 2020), a large scale language model that is trained with 300B tokens from text data sources like Common Crawl corpus (Raffel *et al.*, 2019) (570 GB of data after filtering and cleaning), WebText (Radford *et al.*, 2019), two internet-based book corpora and Wikipedia pages. The datasets had to be curated and processed to provide meaningful learning signals to the training models. While recent advances in self-supervised learning have reduced the dependence on large-scale, clean and well-labeled datasets, we still need to account for the time and cost of pre-training large-scale models. For instance, the *GPT-3* model used compute equivalent to  $3.14e^{23}$  flops<sup>2</sup> and it would take 355 years to train GPT-3 on a single NVIDIA Tesla V100 GPU. The sample efficiency of ML systems significantly lags behind that of humans, making them expensive to develop and deploy.

A second key challenge is that standard AI paradigms are not good at transferring (or leveraging) knowledge across tasks. While it is possible to train systems that provide excellent performance on a specific task (or related distribution of tasks, in the case of multi-task learning), it is much harder to train general-purpose AI systems that can perform a diverse set of tasks. When AI systems are trained over a sequence of tasks, they tend to *forgets* the crucial knowledge they acquired from the

---

<sup>2</sup>floating point operations



previous tasks. This phenomenon is often referred to as catastrophic forgetting (McCloskey and Cohen, 1989; Ratcliff, 1990) and affects all parametric AI systems. Sometimes, knowledge transfer even hurts the performance on the current task due to *negative interference* (a common challenge for multi-task learning) of knowledge across tasks (Standley *et al.*, 2020; Yu *et al.*, 2020; Mansilla *et al.*, 2021; Chen *et al.*, 2018). Even in the case of paradigms like transfer learning (which specifically emphasizes the transfer of knowledge across tasks), the knowledge transfer is often uni-directional, i.e., the knowledge from the previous tasks is used to improve the performance on the current task (and not all the tasks). The emphasis is on improving the performance of the current task, even if that hurts the performance of the previous tasks. In an ideal world, we would want the learning systems to perform both *forward* (training on the current task improves the performance on the future tasks) as well as *backward* transfer of knowledge (training on the current task improves the performance on the previous tasks).

These two challenges are related. AI systems need a lot of data to train on because they start training on every task from *scratch*. Imagine a system that has to learn the alphabet every time it reads a book. Such a system would have a poor sample complexity because it cannot transfer knowledge across tasks (of learning alphabets and reading books). In terms of learning strategy, the current AI systems are closer to this hypothetical system than humans. As the new data becomes available, the AI systems can not *incrementally* acquire new knowledge (without forgetting the prior knowledge). These challenges also make the AI systems harder to adapt to new tasks/datasets. Since these systems do not effectively transfer knowledge across tasks, they need a lot of data to adapt to the new task when they encounter a new task. These behaviors are in sharp contrast to how humans learn and behave. Humans do not need to *train* over a stationary data distribution for multiple epochs. While they do not have perfect memory, they can incrementally acquire and update knowledge over their lifetime without catastrophically forgetting the knowledge relevant for the previous tasks. Moreover, humans can efficiently leverage experience across tasks and exhibit knowledge transfer to improve performance on new (forward transfer) and previous (backward transfer) tasks. Over

time, humans learn how to quickly adapt to novel situations without learning everything from scratch.

The *Lifelong Learning* paradigm is the branch of AI that focuses on developing lifelong learning systems - systems that keep accumulating new knowledge throughout their lifetime without forgetting the prior knowledge and use this accumulated knowledge to improve their performance on the different tasks. We highlight that the lifelong learning paradigm is not unique to the multi-task setup and applies to the single-task setup as well. Lifelong learning is a general setup since it makes fewer assumptions about the task (or tasks). Consider a standard single-task supervised learning setup where the learner can access the entire dataset before starting the training. In this case, the learner can perform multiple epochs over the dataset, shuffling the data in each epoch to keep the data distribution, i.i.d (independent and identically distributed). However, there are many implicit assumptions in this setup - since we have access to the dataset beforehand, we know how many unique classes exist in the dataset. We also have access to the class distribution and can weigh the classes differently. We can also over/under-sample the data. While these assumptions make the setup amenable for training, they also take the setup away from the more general open-ended learning setup. If we were not to assume access to the dataset (or even the number of unique classes), the AI system would have to address challenges like modifying the network architecture as it sees new classes, not forgetting the old data points as it trains on new data points and potentially increasing the capacity of the system as new data keeps coming in. All these challenges are studied under the paradigm of lifelong learning.

## 1.4 Outline

This primer is an attempt to provide a detailed summary of the different facets of lifelong learning. We start with Chapter 2 which provides a high-level overview of lifelong learning systems. In this chapter, we discuss prominent scenarios in lifelong learning (Section 2.4), provide a high-level organization of different lifelong learning approaches (Section 2.5), enumerate the desiderata for an ideal lifelong learning system

(Section 2.6), discuss how lifelong learning is related to other learning paradigms (Section 2.7), describe common metrics used to evaluate lifelong learning systems (Section 2.8). This chapter is more useful for readers who are new to lifelong learning and want to get introduced to the field without focusing on specific approaches or benchmarks.

The remaining chapters focus on specific aspects (either learning algorithms or benchmarks) and are more useful for readers who are looking for specific approaches or benchmarks. Chapter 3 focuses on regularization-based approaches that do not assume access to any data from previous tasks. Chapter 4 discusses memory-based approaches that typically use a *replay buffer* or an *episodic memory* to save subset of data across different tasks. Chapter 5 focuses on different architecture families (and their instantiations) that have been proposed for training lifelong learning systems. Following these different classes of learning algorithms, we discuss the commonly used evaluation benchmarks and metrics for lifelong learning (Chapter 6) and wrap up with a discussion of future challenges and important research directions in Chapter 7.

## 1.5 Scope

The primer is designed to serve as an introduction to lifelong learning paradigm and address questions like “what is lifelong learning”, “why is it a relevant problem to work on”, “what are some key desiderata of a lifelong learning system”, “what are some common design decisions when developing lifelong learning system”, “what are commonly used benchmarks in lifelong learning” etc. While we include (and describe) several lifelong learning approaches and benchmarks and intend to keep the document updated over time, the primer is not an exhaustive literature survey by any means. The selection of work is based on the diversity of approaches and pedagogical reasons. We note that we are focusing on lifelong learning approaches in the context of supervised learning and do not cover work-related to lifelong reinforcement learning, which is an important and interesting topic on its own. We recommend the readers to refer Khetarpal *et al.* (2020) for a survey on lifelong reinforcement learning.

## 1.6 Target Audience

The target audience for this primer is both newcomers (people who are new to the field of lifelong learning or are just curious about lifelong learning) and practitioners (who are working on lifelong learning or related areas like meta-learning, transfer learning, multi-task learning, etc.). It should be useful for people across the spectrum - from researchers working on the fundamental problems to practitioners working on applications of ML. Chapter 2 is particularly useful for readers who are new to the area of lifelong learning. Readers already familiar with lifelong learning may benefit more from Chapter 3, Chapter 4 and Chapter 5 that focus on different classes of lifelong learning algorithms. Readers looking to evaluate their lifelong learning systems or create new evaluation benchmarks would benefit from a discussion on benchmarks and metrics (Chapter 6).

# 2

---

## Overview of Lifelong Learning

---

### 2.1 What is Lifelong Learning

Consider a setup where a machine learning model is trained over a sequence of tasks. Let us assume that the model has trained on the first  $k$  tasks and is starting to train on the  $k + 1^{th}$  task. As the model trains on the  $k + 1^{th}$  task, a couple of scenarios are possible: (i) the model learns to solve the current task at the expense of performance on the previous tasks, (ii) the model fails to learn the new tasks though it retains its performance on the previous tasks, (iii) the model learns the new tasks while retaining its performance on the previous tasks, or (iv) the model does not learn the new task while forgetting its knowledge on the previous task. While the ideal outcome is the one where the model learns the new tasks while retaining its performance on the previous tasks, in practice, the model would likely forget some of the previous knowledge and may not be able to learn the new task.

This setup can be viewed from the lens of **stability-plasticity dilemma** (Mermillod *et al.*, 2013). Here, *plasticity* refers to the ability to integrate new knowledge, and *stability* refers to the ability to retain previous knowledge (Mirzadeh *et al.*, 2020a). Too much plasticity will likely lead to forgetting previous knowledge, while too much stability

will hurt learning on the current task. Any learning system, biological or artificial, needs to balance plasticity with stability to ensure continued learning without catastrophic forgetting.

Much work in machine learning looks at the *stability-plasticity dilemma* as two separate problems and puts more emphasis on one of the two aspects. For example, transfer learning approaches focus exclusively on the plasticity aspect, while approaches to alleviate catastrophic forgetting focus more on the stability aspect. The *Lifelong Learning* paradigm focuses on both the challenges at once, with the goal of developing lifelong learning systems - systems that keep accumulating new knowledge throughout their lifetime (plasticity) without catastrophically forgetting the prior knowledge (stability) and use this accumulated knowledge to improve their performance on the different tasks.

As discussed in Section 1.5, in this primer, we focus on lifelong learning paradigm in context of supervised learning. We briefly recap the supervised learning setup (Section 2.2), describe the lifelong supervised learning paradigm (Section 2.3) and discuss three prominent scenarios in lifelong supervised learning (Section 2.4). For the sake of simplicity, we drop the term *supervised* when referring to lifelong learning and make it explicit when we are referring to lifelong reinforcement learning.

## 2.2 Background: Supervised Learning

In supervised learning, we want to learn a function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that is able to predict a target vector  $y \in \mathcal{Y}$ , when given an input sample  $x \in \mathcal{X}$  (where  $x$  can be in raw form, or in the form of a curated set of features for the raw input). To do so, we have access to some training data  $D = \{(x_i, y_i)_{i=1}^n\}$ , which consists of  $n$  pairs of input samples  $x_i \in \mathcal{X}$  and their corresponding target vectors  $y_i \in \mathcal{Y}$ . We assume data is drawn i.i.d. from a fixed distribution  $P(x, y)$ .

In order to train this function  $f$ , we use some loss function  $L$  that captures how much the function's prediction  $\hat{y} = f(x)$  is different from the ground truth  $y$  given a sample  $x$ . The risk associated with this

function becomes:

$$R(f) = \mathbb{E}_{x,y \sim P}[L(f(x), y)], \quad (2.1)$$

and hence the optimal function  $f^*$  is the function that minimizes this risk:

$$f^* = \arg \min_f R(f). \quad (2.2)$$

However, since the distribution  $P$  is unknown, the risk  $R$  cannot be computed. As an alternative, the Empirical Risk Minimization (ERM) principle (Vapnik, 1991) is usually used, which seeks to obtain the optimal function  $\hat{f}$  that minimizes the empirical risk  $\hat{R}$

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i), \quad (2.3)$$

$$\hat{f} = \arg \min_f \hat{R}(f). \quad (2.4)$$

### 2.3 Lifelong Learning Formulation

In the lifelong learning setup, there exists a sequence of tasks, where each task  $t$  represents a set of unique classes  $\mathcal{C}^{(t)}$ , where  $\mathcal{C}^{(t)} \subseteq \mathcal{Y}$  (the set of all possible classes). The tasks come in a sequence one by one and each task  $t$  comes with its set of data  $D^{(t)} = \{(x_i, y_i)_{i=s}^{s+n_t}\}$ , where  $x_i \in \mathcal{X}$  and  $y_i \in \mathcal{C}^{(t)}$ .

The output space  $\mathcal{Y}^{(\mathcal{T})}$  keeps expanding whenever a new task  $\mathcal{T}$  is introduced  $\mathcal{Y}^{(\mathcal{T})} = \bigcup_{t=1}^{\mathcal{T}} \mathcal{C}^{(t)}$ , the goal is still to learn the function that maps the input to output space across all seen tasks  $f_{\mathcal{T}} : \mathcal{X} \rightarrow \mathcal{Y}^{(\mathcal{T})}$ . Applying the ERM principle as is would lead us to the following equation:

$$\hat{R}_{\mathcal{T}}(f) = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} \frac{1}{|D^{(t)}|} \sum_{(x_i, y_i) \in D^{(t)}} L(f(x_i), y_i), \quad (2.5)$$

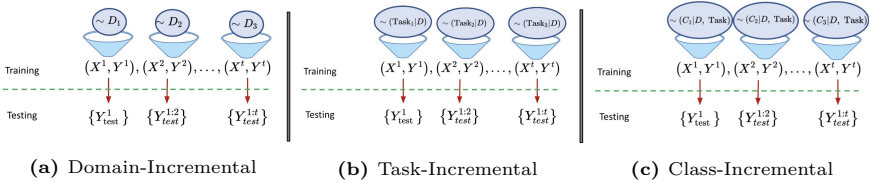
$$\hat{f}_{\mathcal{T}} = \arg \min_f \hat{R}_{\mathcal{T}}(f). \quad (2.6)$$

However, as the data from older tasks  $t < \mathcal{T}$  is not available anymore, calculating the risk this way becomes infeasible. On the other hand,

minimizing the risk on only the currently available data will lead to good performance on the current task and potential catastrophic forgetting of the previous tasks. We shall explain in the next chapter how the different existing methods try to deal with this issue.

## 2.4 Prominent Scenarios in Lifelong Learning

There are three prominent scenarios in lifelong learning: Domain-incremental Learning, Task-incremental Learning, and Class-incremental Learning. These scenarios assume that during training, there are clear and well-defined boundaries between the tasks to be learned (Ven and Tolia, 2019b) (though the learning system may not have access to these task boundaries). These scenarios are distinguished by whether task identity  $t$  is provided during evaluation and, if it is not, whether task identity must be inferred.



**Figure 2.1:** Overview of the three lifelong learning scenarios

### 2.4.1 Domain-incremental Learning

In the domain-incremental learning scenario (Figure 2.1a), the system does not need (and does not have) access to the task identity  $t$  during evaluation. In this setup, the input distributions are different, while the output distribution is the same, i.e.,  $P(x^{(a)}) \neq P(x^{(b)})$  and  $\mathcal{C}^{(a)} = \mathcal{C}^{(b)} \forall a, b \in W$  if  $a \neq b$  where  $W$  is the set of whole numbers. In this setup, the models have a single-headed output layer, and each class has the same semantic meaning across all the tasks. Since the system does not have to choose an output head, it does not need to infer the task identity.



### 2.4.2 Task-incremental Learning

In the task-incremental learning scenario, the model is trained on a sequence of tasks with known task identities. Since task identity is always provided, it is possible to train models with task-specific components using a *multi-headed* output layer (for deep neural networks) (Kirkpatrick *et al.*, 2017; Chaudhry *et al.*, 2019b; Mirzadeh *et al.*, 2020b). The output classes are disjoint between tasks,  $P(x^{(a)}) \neq P(x^{(b)})$ ,  $P(y^{(a)}) \neq P(y^{(b)})$ ,  $\mathcal{C}^{(a)} \cap \mathcal{C}^{(b)} = \Phi \quad \forall a, b \in W$  if  $a \neq b$  in the task-incremental scenario and models are evaluated by their average final performance across all tasks after being trained on all tasks sequentially (see Figure 2.1b). Here, when evaluating on a given task, the model’s predictions for only the classes corresponding to the given task are considered (Ven and Tolias, 2019b).

### 2.4.3 Class-incremental Learning

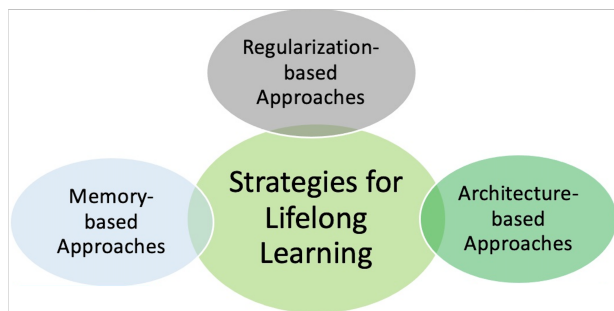
In the class-incremental learning scenario, the model must infer the task identity and solve the tasks seen so far. It is, by far, the most challenging setting in lifelong learning, and many existing methods fail in this setting (Rebuffi *et al.*, 2017; Aljundi *et al.*, 2019b). This scenario employs a single-head architecture where the output space is the same for all distributions, and the model needs to classify all labels without a task-ID (Figure 2.1c). Here,  $P(x^{(a)}) \neq P(x^{(b)})$  and  $P(y^{(a)}) \neq P(y^{(b)}) \quad \forall a, b \in W$  if  $a \neq b$ . For instance, considering a classification task using deep neural networks, the units of all the classes seen so far are active in this scenario.

## 2.5 An overview of Lifelong Learning strategies

A wide range of methods have been proposed in the past years to tackle the challenges in lifelong learning. However, each method makes assumptions that are not consistent due to the presence of different settings defined above. In particular, a few methods require fewer supervisory signals during both training and inference times and hence generalize better to different lifelong learning settings. Such signals can be a natural number for task identity, a natural language descriptor, a

vector representation of data describing a task, etc. However, there is a clear trend in recent works to simultaneously apply multiple techniques to tackle this problem.

Methods proposed in lifelong learning are broadly categorized into the following three categories: Regularization-based, Memory-based, and Architecture-based methods (De Lange *et al.*, 2019; Masana *et al.*, 2020a).



**Figure 2.2:** Common strategies in Lifelong Learning

### 2.5.1 Regularization-Based Methods

Regularization-based methods prevent a drastic change in the network parameters as the new task arrives to mitigate forgetting. These methods are further classified as importance-based, Bayesian-based, distillation-based, and optimization trajectory-based. Here, the importance-based methods regularize the loss function to minimize changes in the parameters important for previous tasks. Distillation-based methods transfers knowledge from the model trained on the previous task to the model being trained on the new data. On the other hand, optimization trajectory-based methods exploit the geometric nature of the local minima to prevent catastrophic forgetting. These methods are shown to be vulnerable to domain shift between tasks (Aljundi *et al.*, 2017). We discuss regularization-based methods in Chapter 3.

### 2.5.2 Memory-Based Methods

Memory-based methods maintain an ‘episodic memory’, containing a few examples from past tasks that are revisited while learning a new task. These methods apply gradient-based updates that facilitate a high-level transfer across different tasks through the examples from the past tasks that are simultaneously available while training on the current new task. For instance, Averaged Gradient Episodic Memory (*AGEM*) (Chaudhry *et al.*, 2019a) uses the episodic memory to project the gradients based on hard constraints defined using the episodic memory and the current mini-batch. Experience Replay (*ER*) (Chaudhry *et al.*, 2019b) uses both replay memory and input mini-batches in the optimization step by averaging their gradients to mitigate forgetting. However, selecting which examples to store is also a significant challenge that has been the focus of various research works. Instead of storing raw samples, Generative Replay trains a deep generative model such as GAN (Goodfellow *et al.*, 2020) to generate data that mimic past data for replay. However, it takes a long time to train such generative models and hence is not a viable option for complex datasets in terms of computational cost. We discuss memory-based methods in Chapter 4.

### 2.5.3 Architecture-Based Methods

Architecture-based methods either freeze or add a set of parameters with the idea that different tasks should have their own set of isolated parameters. These methods alleviate catastrophic forgetting in general, but they rely on a strong base network and work on a small number of tasks. For example, Aljundi *et al.* (2017) assigns a model copy to every new task that arrives. Similarly, there are expansion-based methods that handle the lifelong learning problem by expanding the model capacity in order to adapt to new tasks (Sodhani *et al.*, 2018; Rao *et al.*, 2019). We discuss architecture-based methods in Chapter 5.

## 2.6 Desiderata of Lifelong Learning Systems

Several works have outlined useful properties and open challenges for lifelong learning systems, both in the context of supervised learning (Sodhani *et al.*, 2021; Veniat *et al.*, 2021; Hadsell *et al.*, 2020) and reinforcement learning (Schaul *et al.*, 2018). We compile these properties into a list of desired properties of a model suitable for lifelong learning settings:

1. **Knowledge Retention** - As the model trains over the new tasks, it should not forget the knowledge from the previous tasks. Learning new tasks should not happen at the expense of the knowledge from the previous tasks. Much of the existing literature focuses on this problem (under the name of Catastrophic Forgetting (McCloskey and Cohen, 1989; French, 1999)). Due to this problem, conventional deep learning tends to focus on offline training, with i.i.d. sampling of mini-batches with multiple epochs over the training data. Therefore, the model requires a significant amount of previous tasks data to learn and accumulate explicit knowledge. Some works refer to this knowledge retention property as *plasticity* or *stability*.
2. **Knowledge Transfer** - The model should be able to reuse the knowledge across tasks. This includes both *forward* transfer of knowledge where the knowledge acquired during previous tasks is used to solve the subsequent tasks, and *backward* transfer where the knowledge acquired in the current/future tasks is used to improve performance on the previous tasks. The underlying premise is, if the tasks are related, this knowledge transfer could lead to faster learning and better generalization. Most current approaches for knowledge transfer focus on the forward transfer of knowledge.
3. **Model Expansion** - As the model trains over a sequence of tasks, the model should be able to *expand* itself or increase its learning *capacity*. This could mean that the model can introduce new trainable parameters in practice. Also, training a separate

model entirely for each task discounts the possibility of transferring knowledge forward and backward directions when the tasks are related. This further discounts better generalization or faster learning. In Section 5.2.3, we discuss expanding networks that aim to tackle these problems by increasing the model capacity and reusing learned representations.

4. **Parameter Efficiency** - While increasing the model's capacity, we would also want the computational and memory costs of the model to increase only sub-linearly (or to be bounded) as the model trains on new tasks to avoid computational performance degradation. The model expansion property comes with additional constraints: In the true lifelong learning setting, the model would experience a continual stream of training data that can not be stored. Hence the model would, at best, have access to only a small sample of the historical data. We can not rely on past examples to train the expanded model from scratch in such a setting, and a zero-shot knowledge transfer is desired.

## 2.7 Relation to Other Areas

Lifelong learning is referred by different names in the literature: incremental learning (Solomonoff, 1989), continual learning (De Lange *et al.*, 2019), explanation-based learning (Thrun, 1996; Thrun, 2012), never-ending learning (Carlson *et al.*, 2010), etc. The underlying idea in all these works is that lifelong learning systems would be more effective at learning and retaining knowledge across different tasks. In principle, the ability to generalize is one of the most important characteristics of a machine learning model. If tasks are related, then knowledge transfer between tasks should lead to a better generalization, and faster learning (Biesialska *et al.*, 2020).

Lifelong learning also bears some resemblance to other dominant research areas. It is closely related to areas like Multitask Learning (Caruana, 1997), Meta Learning (Schmidhuber, 1987; Thrun and Pratt, 1998), Transfer Learning (Pan and Yang, 2009), Online Learning (Shalev-Shwartz and Singer, 2007; Shalev-Shwartz, 2012), and Curriculum Learn-

ing (Bengio *et al.*, 2009).

### 2.7.1 Multitask Learning

The paradigm of multitask learning focuses on improving the performance of a single model on multiple tasks by sharing knowledge across tasks (Caruana, 1997; Zhang *et al.*, 2014; Ruder, 2017; Radford *et al.*, 2019; Sodhani *et al.*, 2021). This goal is quite similar to the goal of lifelong learning systems, with one major difference - multitask learning approaches generally assume that information about all the tasks is known when the training starts. In practice, this means that the learning system has access to all the tasks, and in some cases, the system can even choose the ordering of the tasks (Bengio *et al.*, 2009; Pentina *et al.*, 2015) as done in curriculum learning. This assumption is generally not valid for lifelong learning setups where neither the number of tasks nor the nature of tasks is assumed to be known when starting the training.

Since the learning system has upfront knowledge about all the tasks, catastrophic forgetting is not usually studied in multitask learning. However, a closely related challenge that is well-studied in the context of multitask learning is the problem of negative interference (Du *et al.*, 2018; Suteu and Guo, 2019; Yu *et al.*, 2020) where the gradients corresponding to the different tasks interfere negatively with each other, thus slowing down (or completely inhibiting) training on multiple tasks. Negative interference is related to catastrophic forgetting as it can cause the learning system to forget (or *unlearn*) knowledge from one or more tasks.

In some cases of multitask learning, referred to as sequential multitask learning (Zhang and Yang, 2017; Xiong *et al.*, 2018), the different tasks may be introduced sequentially, over a period of time. This setup is closer to the lifelong learning setup (as compared to the general multitask learning setup) but even in the case of sequential multitask learning, the information about all the tasks is generally assumed to be known upfront.

Multitask Learning also shares several similarities with lifelong learning in terms of inductive biases and architecture choices. For example, modular networks are a common design choice for both multitask learn-

ing (Liu *et al.*, 2019; Devin *et al.*, 2017; Chang *et al.*, 2019; Sodhani *et al.*, 2021) and lifelong learning (Section 5.1). In both the cases, the inductive bias of compositionality and learning expert *knowledge* (or *skills*) is seen as a useful property for the learning model.

### 2.7.2 Meta Learning

Meta Learning, also known as *learning to learn* (Thrun and Pratt, 1998; Bengio *et al.*, 2013), is the machine learning paradigm that focuses on enabling the training system to learn aspects of the learning process itself. This paradigm can be seen as a natural extension from learning features and models to learning *algorithms*. Meta Learning comprises of three broad family of approaches: (i) Metric-based (Koch, 2015; Vinyals *et al.*, 2016; Sung *et al.*, 2018; Snell *et al.*, 2017), (ii) Model-based (Santoro *et al.*, 2016; Munkhdalai and Yu, 2017), and (iii) Gradient-based (Mishra *et al.*, 2018; Ravi and Larochelle, 2017; Finn *et al.*, 2017).

Meta-learning and lifelong learning approaches have similar motivation - train on the distribution of tasks to improve performance on new, potentially unseen tasks. Similar to lifelong learning, several meta-learning approaches generally do not assume access to all the training tasks at the start of the training. Several works have started focusing on the intersection of lifelong learning and meta-learning (Al-Shedivat *et al.*, 2018; Ritter *et al.*, 2018b; Nagabandi *et al.*, 2019; Javed and White, 2019; Wang *et al.*, 2020b). However, the two paradigms also have some differences. Unlike lifelong learning methods, Meta-learning approaches generally do not focus on challenges like catastrophic forgetting or capacity saturation. On the other hand, meta-learning approaches use an explicit objective function that incentives faster training on the new tasks while lifelong learning implicitly optimizes for accelerating training.

### 2.7.3 Transfer Learning

The paradigm of transfer learning (Dai *et al.*, 2009; Pan and Yang, 2009; Torrey and Shavlik, 2010; Bengio, 2012; Weiss *et al.*, 2016; Ying *et al.*, 2018; Tan *et al.*, 2018; Zamir *et al.*, 2018; Zhuang *et al.*, 2020) focuses on transferring knowledge from one or more *source* tasks to

one or more *target* tasks. It is related to the lifelong learning paradigm as the learning system is trained over multiple tasks with the hope of doing a forward knowledge transfer to the subsequent tasks.

Transfer learning faces several challenges similar to lifelong learning when considering to transfer a trained model to a new task: (i) Should the model’s architecture be changed (for example by adding more parameters (Rusu *et al.*, 2016) or modules (Auda and Kamel, 1999))? (ii) Should some parts of the model be frozen (if yes, which parts?) or should the entire network be finetuned? (iii) How should we set the learning rate on the new tasks? (iv) How to **infer** the *relatedness* between the tasks. Interestingly, some of the architecture choices and inductive biases (like modular architectures) that are useful for lifelong learning (Veniat *et al.*, 2021) is useful for transfer learning as well (Houlsby *et al.*, 2019; Stickland and Murray, 2019).

However, transfer learning is also different from lifelong learning in several ways: (i) Transfer learning generally focuses on *one-way* transfer of knowledge, from the older to the newer task. In contrast, lifelong learning focuses on *two-way* transfer of knowledge, from both old tasks to new tasks and vice-versa. (ii) Transfer learning focuses primarily on the performance of the current task. Catastrophic forgetting is not seen as a problem and is often not even measured. On the other hand, lifelong learning aims to improve performance over all the tasks. (iii) Transfer learning is often used to *initialize* a model such that it can perform well on the target task. Hence, many approaches for transfer learning involve pre-training on a large corpus. This is not the case with Lifelong learning.

#### 2.7.4 Online Learning

Standard machine learning paradigms (especially in the context of supervised learning and unsupervised learning) use the *batch* (or *offline*) learning approach where any given data point can be used for training any number of times. While this approach often works well in practice, it may be infeasible in certain setups. For example, there may be privacy-related restrictions for storing the data, making it infeasible to use it for offline training. In other cases, the size of the training



data could be unbounded (as in the case of click-stream data). In such a case, storing (and training over) all the historical data is not practical. *Online learning* (Shalev-shwartz and Singer, 2007; Shalev-Shwartz, 2012; Zinkevich, 2003) is a paradigm in machine learning that aims to address some of these limitations.

Online learning techniques have been used in conjunction with other machine learning paradigms like multi-task learning (Dekel *et al.*, 2006; Agarwal *et al.*, 2008; Li *et al.*, 2013; Wang *et al.*, 2016), metric learning (Shalev-Shwartz *et al.*, 2004; Jain *et al.*, 2008; Wu *et al.*, 2016), transfer learning (Zhao *et al.*, 2014; Zhao *et al.*, 2011; Ge *et al.*, 2013; Bhatt *et al.*, 2012) etc. The connection between lifelong learning and online learning is less obvious as the majority of works in lifelong learning focus on the *batch* (samples within a task) setup. However, several recent lifelong learning works are starting to focus on the online setup and have argued in favor of online lifelong learning setup to be closer to real-life learning as compared to the offline counterpart (Chrysakis and Moens, 2020; Sodhani *et al.*, 2020; Parisi and Lomonaco, 2020; Mai *et al.*, 2021; Aljundi *et al.*, 2019a; Aljundi *et al.*, 2019c; Pham *et al.*, 2021; Liu, 2020; Kruszewski *et al.*, 2021; Malviya *et al.*, 2021).

### 2.7.5 Curriculum Learning

Humans and animals learn more efficiently when starting with simpler *concepts* (or tasks) and progressively learning more complex concepts (or tasks) (Skinner, 1958; Peterson, 2004). For example, the human education system is designed as a curriculum where new concepts build on (and leverage) previous concepts. Curriculum learning (Elman, 1993; Bengio *et al.*, 2009; Hacoheh and Weinshall, 2019; Wang *et al.*, 2020a) is the machine learning paradigm that aims to leverage insights about the importance of curriculum and use these insights to improve the training of machine learning models. Given the generic nature of curriculum learning, it has been used in conjunction with other machine learning paradigms like multi-task learning (Pentina *et al.*, 2015; Sarafianos *et al.*, 2017; Murugesan and Carbonell, 2017), reinforcement learning (Narvekar, 2017; Narvekar and Stone, 2018; Narvekar *et al.*, 2020; Portelas *et al.*, 2020), transfer learning (Dong *et al.*, 2017; Weinshall

*et al.*, 2018), etc.

Curriculum learning and lifelong learning have several commonalities. Both the paradigms can be motivated from the perspective of human cognition and involve a notion of *continuous* learning - over a sequence of datasets in lifelong learning and over a sequence of splits of one (or more) datasets in curriculum learning. However, there are notable differences as well. In the general lifelong learning setup, the learning system has no control over the sequence of data points. In contrast, curriculum learning focuses on the most optimal sequence of data points (optimal to learning). In curriculum learning, information about the different datasets/data points is available beforehand, while in the lifelong learning setup, this information becomes available during training. Despite these differences, curriculum learning can be a helpful technique in the context of lifelong learning, and the general idea of selecting data points, based on their estimated *hardness*, has been used in several approaches for experience replay (Andrychowicz *et al.*, 2017; Li and Ji, 2021).

## 2.8 Common Metrics in Lifelong Learning

Lifelong learning differs from supervised learning in terms of how the systems are trained and evaluated. These differences imply that the use of the traditional, *single-task* performance metrics like top-1 or top-5 error rates is not suitable for lifelong learning systems. As discussed in the previous sections, alleviating catastrophic forgetting and knowledge transfer are the crucial challenges that the methods should focus on in lifelong learning. Therefore, we need metrics to measure the models' performance appropriately in a lifelong learning setup. The metrics should evaluate lifelong learning methods to assess their performance through time, including how much the model forgets or gains on the previously learned knowledge. In this section, we explain some of the most popular metrics in lifelong learning, including the average accuracy for overall performance (Chaudhry *et al.*, 2019a), the average forgetting (Chaudhry *et al.*, 2018), the forward and the backward knowledge transfer that assesses the ability of the models to transfer knowledge (Lopez-Paz and Ranzato, 2017a; Lesort *et al.*, 2019).

### 2.8.1 Performance Metrics

In lifelong learning settings, a system learns from the dataset  $(x_1, y_1), \dots, (x_{\mathcal{T}}, y_{\mathcal{T}})$ , at each episode where  $x_i$  denotes input variable and  $y_i$  denotes target/output variable belonging to training set of a task  $i$ . However, the system’s performance is reported based on  $\{x_1^{test}, y_1^{test}\}, \dots, \{x_{1:\mathcal{T}}^{test}, y_{1:\mathcal{T}}^{test}\}$ .

In the class incremental learning setting, the system incrementally learns a set of classes. The model also incrementally learns a new task at each time in task incremental learning. The model aims to have less forgetting through time and better performance. The **average accuracy** (Chaudhry *et al.*, 2018) is computed as follows:

$$A_{\mathcal{T}} = \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} a_{\mathcal{T},i}, \quad (2.7)$$

where  $A \in [0, 1]$ ,  $\mathcal{T}$  is the total number of tasks seen so far, and  $a_{n,i}$  is the test classification accuracy on task  $i$  after sequentially learning the  $n^{\text{th}}$  task. Forgetting Measure is the another metric that is very crucial in the lifelong learning model performance report. Chaudhry *et al.* (2018) introduce the **forgetting** measure, formally defined as follows:

$$F_{\mathcal{T}} = \frac{1}{\mathcal{T} - 1} \sum_{i=1}^{\mathcal{T}-1} f_{\mathcal{T},i}, \quad (2.8)$$

where  $F \in [-1, 1]$ ,  $f_{j,i}$  is a measure of forgetting on task  $i$  after training up to task  $j$ .  $f_{j,i}$  is defined as the difference between best accuracy achieved on task  $i$  in the past and the final accuracy of task  $i$  after training on task  $j$ :

$$f_{j,i} = \max_{k \in \{1, \dots, j-1\}} a_{k,i} - a_{j,i}. \quad (2.9)$$

The **average forgetting ratio** is another metric introduced by Serra *et al.* (2018). It measures the amount of forgetting over time and studies the effectiveness of the lifelong learning method in multiple datasets relatively. After training on task  $t$ , it computes the accuracy on all testing sets of tasks  $\tau \leq t$ . This process is repeated multiple times

using different seeds for uniformly randomized task-order. Then, the forgetting ratio is defined as follows:

$$\rho^{\tau \leq t} = \frac{A^{\tau \leq t} - A_{\text{R}}^{\tau}}{A_{\text{J}}^{\tau \leq t} - A_{\text{R}}^{\tau}} - 1, \quad (2.10)$$

where  $A^{\tau \leq t}$  is the accuracy measured on task  $\tau$  after sequentially learning task  $t$ ,  $A_{\text{R}}^{\tau}$  is the accuracy of a random multi-layer linear classifier using the class information of task  $\tau$  and  $A_{\text{J}}^{\tau \leq t}$  is the accuracy measured on task  $\tau$  after jointly learning  $t$  tasks in a multitask learning manner (Serra *et al.*, 2018). To compute the average ratio, we can simply compute the average as follows:

$$\rho^{\leq t} = \frac{1}{t} \sum_{\tau=1}^t \rho^{\tau \leq t}. \quad (2.11)$$

The Positive Backward Transfer and Forward Transfer metrics are two more important metrics in lifelong learning. Mai *et al.* (2021) visually shows how we can compute these metrics and what they measure. Figure 2.3 illustrates their explanation.

a	$te_1$	$te_2$	...	$te_{T-1}$	$te_T$
$tr_1$	$a_{1,1}$	$a_{1,2}$	...	$a_{1,T-1}$	$a_{1,T}$
$tr_2$	$a_{2,1}$	$a_{2,2}$	...	$a_{2,T-1}$	$a_{2,T}$
...	...	...	...	...	...
$tr_{T-1}$	$a_{T-1,1}$	$a_{T-1,2}$	...	$a_{T-1,T-1}$	$a_{T-1,T}$
$tr_T$	$a_{T,1}$	$a_{T,2}$	...	$a_{T,T-1}$	$a_{T,T}$

**Figure 2.3:**  $tr_i$  and  $te_i$  denote training and test set of task  $i$ .  $BWT^+$  is the average of the difference between accuracies in purple and accuracies in the diagonal.  $FWT$  is the average of accuracies in green.  $A_T$  is the average of accuracies in the box in the last row (Mai *et al.*, 2021).

The Positive Backward Transfer metric measures the positive influence of learning a new task on preceding tasks' performance. Positive Backward Transfer metric is denoted as  $BWT^+$  and computed as fol-

lows (Mai *et al.*, 2021):

$$\begin{aligned}
 BWT &= \frac{\sum_{i=2}^T \sum_{j=1}^{i-1} (a_{i,j} - a_{j,j})}{\frac{T(T-1)}{2}} \\
 BWT^+ &= \max \left( \frac{\sum_{i=2}^T \sum_{j=1}^{i-1} (a_{i,j} - a_{j,j})}{\frac{T(T-1)}{2}}, 0 \right)
 \end{aligned} \tag{2.12}$$

where Backward Transfer and Positive Backward Transfer are denoted as  $BWT$  and  $BWT^+$  respectively. As Figure 2.3 shows, the purple area corresponds to the area used to compute Positive Backward Transfer.  $BWT < 0$  indicates catastrophic forgetting, and  $BWT > 0$  indicates that learning new tasks has helped with the preceding tasks (Ebrahimi *et al.*, 2020). The Forward Transfer metric denoted as  $FWT$  measures the positive influence of learning a task on future tasks' performance. We can compute  $FWT$  as follows:

$$FWT = \frac{\sum_{i=1}^{j-1} \sum_{j=1}^T a_{i,j}}{\frac{T(T-1)}{2}} \tag{2.13}$$

In Figure 2.3,  $FWT$  is the average of accuracies in green.

Learning Curve Area ( $LCA \in [0, 1]$ ) is another performance metric proposed by Chaudhry *et al.* (2019a). To explain the LCA metric, we need to define an average  $b$ -shot performance after the model has been trained for all the  $T$  tasks as:

$$Z_b = \frac{1}{T} \sum_{k=1}^T a_{k,b}, \tag{2.14}$$

where  $b$  is the number of mini-batches. LCA at  $\beta$  is defined as the area of the convergence curve  $Z_b$  as a function of  $b \in [0, \beta]$ :

$$LCA_\beta = \frac{1}{\beta + 1} \int_0^\beta Z_b db = \frac{1}{\beta + 1} \sum_{b=0}^\beta Z_b \tag{2.15}$$

It is worth mentioning that  $LCA_0$  is the average zero-shot performance and is considered the same as the forward transfer performance.  $LCA_\beta$  and the area under the  $Z_b$  curve will be high when the zero-shot performance is good, and it shows how quickly the model learns new tasks.

This metric is valuable when two models have the same  $Z_\beta$  or  $A_T$ , but very different  $LCA_\beta$  where one learns much faster than the other with same final accuracy (Chaudhry *et al.*, 2019a).

Aside from the metrics discussed here, there are other useful metrics that can reveal the potential weakness or strength of the methods. Following is a list of some of the traditional performance metrics that can be used in the lifelong learning domain.

- Throughput (images/sec) at train and test time.
- Mean and standard deviation of top-1 and top-5 error rates for each individual task in task incremental learning.
- Comparison of the method’s performance considering the replay buffer size or the memory overhead.
- Confusion matrix comparison. Since it is tough to observe the differences between the reported confusion matrix from different approaches, we have to find a nice and cheap way to compare two very similar confusion matrix (Wu *et al.*, 2019b; Hou *et al.*, 2019; Abdelsalam *et al.*, 2021).
- Similarity Measurement of the classes that the model should learn continually. In this case, the order of the tasks will be more meaningful in the experiment result. Since the forgetting of the model correlates with the order of the task, considering the tasks sequence and measuring their similarity will be essential to have a fair comparison with other methods.

### 2.8.2 Time

The time-related metrics that can be reported either task by task or as a historical average are task’s training time comparison, testing time comparison, and validation time comparison.

### 2.8.3 Memory

Replay-based methods are the most popular methods in lifelong learning. These methods do not assume any limitations to access to data from

the previous tasks. In practice, samples of data, from the previous tasks, are retained into a memory bank called the *replay buffer*. The performance of these methods depends on the size of the replay buffer and the strategy of selecting, editing, and removing samples from the replay buffer. Indeed, the size of the memory or replay buffer is the most critical parameter that should be reported and included in the model performance evaluation process. Following is a list of simple metrics and parameters that we should consider when comparing methods.

- Replay buffer size and the strategy for constructing and updating the replay buffer.
  - Fixed memory size should be reported if a fixed window is used to keep samples in the replay.
  - Some methods use a replay such that a fixed window per class is reserved to keep track of replayed samples. In this case, there is no fixed memory size for the lifelong learning method, but instead, a fixed window is reserved for each class. Therefore as the model learns new classes or tasks, we expand the memory size for new tasks or set of classes. Reporting the strategy that is used to construct the memory and the size of the memory is important.
- A metric to show how much the memory consumption grows as the model learns a new concept. This is different from the size of the replay that is explained above. Some approaches need more memory to alleviate the catastrophic forgetting and use the parameter isolation method (discussed in Section 5.2). In such cases one needs to keep track of the memory consumption overhead as models learn new concepts through time.
- A metric to evaluate the memory update rates when we train a lifelong learning model in a large-scale memory-based distributed computing cluster. The memory update rates are important in such cases because of the network communication overhead that might decrease the computation speed up either at training or testing time.

- The number of memory components also should be reported if the proposed method uses a different type of extra memory component for different purposes such as keeping previous task models or hidden representation or extracted features of some samples from the previous task.



# 3

---

## Regularization-based Approaches

---

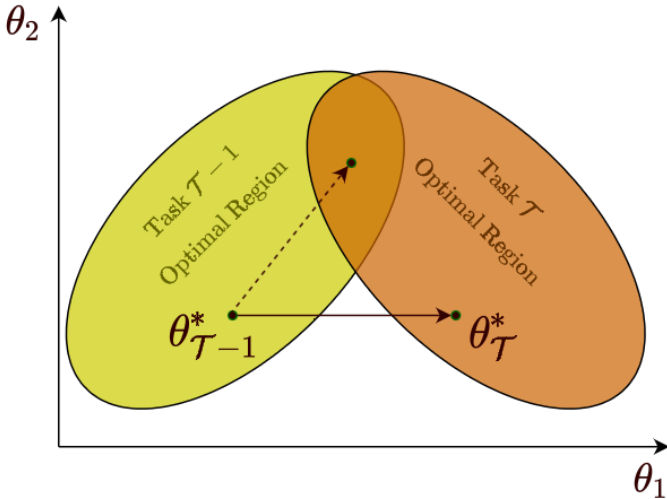
In this chapter, we shall explore how the different methods try to overcome the challenges of lifelong learning without having access to any data from previous tasks nor having the ability to expand the network to learn new tasks. It should be noted that this chapter and the following two chapters are, for the most part, orthogonal to each other in their approaches, which means that the different approaches can, in practice, be combined as done in previous works like Sodhani *et al.* (2020).

When a neural network is trained sequentially on a series of tasks, the network parameters try to minimize the objective on the current task irrespective of what happens to the performance of the previous tasks. In other words, the network does not see except what we allow it to see, and it does not solve except what we ask it to solve. When the network is trained on task 1, we ask it to minimize the objective on task 1. However, when it is trained on task 2, if we only ask it to minimize the objective on task 2 (as we do not have access to the data of task 1 anymore), this can lead to the deterioration of the network performance on task 1. The reason behind this is that task 1 is not incorporated in the objective function when training on task 2. This kind of behavior

is known as catastrophic forgetting (see Figure 3.1), which takes place when the parameters of the network change sufficiently across the tasks, such that their performance on previous tasks degrades significantly.

### 3.1 Definition

Let us consider a setup where we receive a continuum of data from different tasks in a sequential manner:  $(x_1, d_1, y_1), \dots, (x_t, d_t, y_t), \dots, (x_{\mathcal{T}}, d_{\mathcal{T}}, y_{\mathcal{T}})$  where  $x_t$  is the input data,  $d_t$  is the task descriptor and  $y_t$  is the target variable of task  $t$ . Let the current task be  $\mathcal{T}$ , and the set of weights after training on task  $\mathcal{T}$  be  $\theta_{\mathcal{T}}$ .  $\theta$  consists of  $N$  parameters  $\theta \in \mathbb{R}^N$ , with  $\theta_t$  denoting the parameters after training on task  $t$ . The training objective  $\tilde{L}$  would normally be our original objective  $L_{\mathcal{T}}$  (for example, cross entropy, in the case of classification, on the data that belongs to



**Figure 3.1:** After learning the task  $\mathcal{T}-1$ , the parameters are at  $\theta_{\mathcal{T}-1}^*$ . While learning the task  $\mathcal{T}$ , if the model follows the solid line to reach  $\theta_{\mathcal{T}}^*$ , it may incur a significant loss on task  $\mathcal{T}-1$ , i.e., it suffers from catastrophic forgetting. On the other hand, the goal is to follow the dashed line to reach an optimal parameter setting to incur a minimal loss on both tasks.

task  $\mathcal{T}$ ).

$$\tilde{L}(\theta) = L_{\mathcal{T}}(\theta) := \frac{1}{|D_{\mathcal{T}}|} \sum_{(x,y) \sim D_{\mathcal{T}}} l(\theta; x, y), \quad (3.1)$$

where  $l$  is the per sample loss, and  $D_{\mathcal{T}}$  is the set of samples that belong to task  $\mathcal{T}$ .

Regularization-based approaches constrain the update of neural networks to prevent catastrophic forgetting by adding a penalty term ( $R_{\mathcal{T}}$ ) such that the new objective function looks like:

$$\tilde{L}(\theta) = L_{\mathcal{T}}(\theta) + R_{\mathcal{T}}. \quad (3.2)$$

Regularization-based approaches can be roughly categorized into four main types: (i) importance-based regularization, (ii) Bayesian-based regularization, (iii) distillation-based regularization, and (iv) optimization trajectory-based regularization. Out of these four categories, the first three categories explicitly define  $R_{\mathcal{T}}$  using old parameters, training examples/outputs or parameter distribution, etc. On the other hand, optimization trajectory-based regularization exploits the geometric nature of the local minima, and the corresponding trajectories followed to prevent catastrophic forgetting. We discuss each of these categories in detail in this chapter.

## 3.2 Importance-Based Regularization

Importance-based Regularization tries to introduce solutions that do not require access to samples from previous tasks to alleviate catastrophic forgetting. The first such solution is to try to make the network parameters after training on task 2 (let us call them  $\theta_2$ ) as close as possible to the parameters of the network trained on task 1 ( $\theta_1$ ). One way to do so would be by applying a quadratic constraint between each pair of parameters in  $\theta_1$  and  $\theta_2$ . Although this solution might seem to significantly limit the capacity of the model to solve the different tasks, it can be supported by the over-parametrization of neural networks, where many configurations of  $\theta$  can still lead to the same performance (Hecht-Nielsen, 1992; Sussmann, 1992). However, doing the regularization this way can still significantly reduce the network’s capacity, especially when training takes place sequentially. It is not guaranteed

that there exists another solution in the vicinity of  $\theta_1$  that can still perform well on both tasks.

More sophisticated solutions under the importance-based regularization umbrella try to solve this problem by making the regularization more selective, giving the network some freedom to change some parameters while limiting this freedom for other parameters, based on the importance of each parameter with respect to the previous tasks. Here the central question becomes how to measure the importance of each parameter in an efficient and tractable way.

Let us make the above ideas more concrete. The objective function defined in Eq. 3.1 has no guarantees on the performance over the data of the previous tasks  $1 : \mathcal{T} - 1$ . Hence, a naive solution would be to add a quadratic constraint so that the parameters when training on task  $\mathcal{T}$  do not deviate from the parameters of task  $\mathcal{T} - 1$ , i.e.,

$$R_{\mathcal{T}} = \alpha \sum_{i=1}^N (\theta_i - \theta_{\mathcal{T}-1,i})^2, \quad (3.3)$$

where  $\alpha$  is a regularization weight. The idea here is to find a new set of weights  $\theta_{\mathcal{T}}^*$  that can perform well on task  $\mathcal{T}$ , while at the same time is close enough to  $\theta_{\mathcal{T}-1}^*$  so that the performance on task  $\mathcal{T} - 1$  is preserved. However, adding this constraint this way might be too limiting to the network’s capacity. Given that not all the parameters  $\theta_i$  contribute equally to the performance, a more selective approach can be used so that the parameters that are more important for previous tasks are regularized more than the other parameters:

$$R_{\mathcal{T}} = \alpha \sum_{i=1}^N \Omega_i^{\mathcal{T}-1} (\theta_i - \theta_{\mathcal{T}-1,i})^2, \quad (3.4)$$

where  $\Omega_i^{\mathcal{T}-1}$  represents the importance of each parameter  $\theta_i$  after training on task  $\mathcal{T} - 1$ . Given that  $\Omega_i^{\mathcal{T}-1}$  is the only measure of importance used, it should also be a function of  $\Omega_i^t \forall t < \mathcal{T} - 1$ , so that the performance is preserved on all the previous tasks  $t < \mathcal{T}$ . In the next section, we shall explore how Elastic Weight Consolidation (*EWC*) (Kirkpatrick *et al.*, 2017) tries to estimate these importance parameters  $\Omega_i$ .

### 3.2.1 Elastic Weight Consolidation (EWC)

*EWC* (Kirkpatrick *et al.*, 2017) uses the diagonal terms of the Fisher information matrix as a proxy for the importance of the parameters ( $\Omega_i$ ). However, it motivates it from a probabilistic perspective. Given a set of labelled data  $\mathcal{D}$ , the optimum set of weights  $\theta$  would be the weights that maximize the posterior probability  $p(\theta|\mathcal{D})$ , where  $p(\theta|\mathcal{D})$  is:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \quad (3.5)$$

Assuming that the dataset  $\mathcal{D}$  is divided into two parts,  $D_A$  and  $D_B$ , where both are independent, then:

$$\begin{aligned} p(\theta|\mathcal{D}) &= \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(D_B, D_A|\theta)p(\theta)}{p(D_B, D_A)} = \frac{p(D_B|\theta)p(D_A|\theta)p(\theta)}{p(D_B)p(D_A)} \\ &= \frac{p(D_B|\theta)}{p(D_B)} \frac{p(D_A|\theta)p(\theta)}{p(D_A)} = \frac{p(D_B|\theta)p(\theta|D_A)}{p(D_B)} \end{aligned} \quad (3.6)$$

By applying  $\log$  to the previous function we get:

$$\log p(\theta|\mathcal{D}) = \log p(D_B|\theta) + \log p(\theta|D_A) - \log p(D_B). \quad (3.7)$$

This means that maximizing the posterior on the dataset  $\mathcal{D}$  is equivalent to maximizing the posterior on the subset  $D_A$  while minimizing the Negative Log-Likelihood (NLL) of  $D_B$  ( $-\log p(D_B|\theta)$ ).

Since calculating the posterior  $p(\theta|D_A)$  is intractable, *EWC* approximates the posterior as a Gaussian distribution with the mean given by  $\theta_A^*$  (the parameters that minimize the NLL on  $D_A$ ), and the diagonal precision given by the diagonal of the Fisher information matrix  $F$ , which is known as the Laplace approximation (MacKay, 1992). Hence, optimizing the parameters  $\theta$  for the posterior in Eq. (3.7) would be equivalent to minimizing the loss:

$$\tilde{L}(\theta) = L_B(\theta) + \alpha \sum_{i=1}^N F_i(\theta_i - \theta_{A,i}^*)^2, \quad (3.8)$$

where  $N$  is the total number of parameters ( $|\theta|$ ) and  $L_B(\theta)$  is the loss on task  $B$  data. *EWC* uses the diagonal terms of the Fisher information matrix, which is equivalent to the positive semi-definite second-order

derivative of the loss near a minimum, as a proxy for the importance of each of the weight. The Fisher information matrix is:

$$F = E[\nabla_{\theta} \log f(x; \theta) \nabla_{\theta} \log f(x; \theta)^T]. \quad (3.9)$$

From Eq. (3.4), we have:

$$\Omega_i^{(\mathcal{T})} = F_i^{(\mathcal{T})} = E[\nabla_{\theta_i} \log f(x; \theta)^2 | \theta_{\mathcal{T}, i}^*], \quad (3.10)$$

where the Fisher information matrix  $F$  is calculated as a point estimate at the end of each task.

When training on subsequent tasks, *EWC* tries to minimize the distance from previous optimal parameters that correspond to each task  $t < \mathcal{T}$ , where the objective function becomes:

$$\tilde{L}(\theta) = L_{\mathcal{T}}(\theta) + \alpha \sum_{t < \mathcal{T}} \sum_i F_i^t (\theta_i - \theta_{t,i}^*)^2. \quad (3.11)$$

Therefore *EWC* requires keeping all the Fisher matrices from previous tasks  $F^t$ , as well as the optimal parameters for these tasks  $\theta_t^*$ , which gives it a memory complexity that increases linearly with the number of tasks  $\mathcal{T}$  ( $\mathcal{O}(N\mathcal{T})$ ). Algorithm 1 shows the pseudo code for *EWC*.

---

**Algorithm 1** EWC

---

**Input:**  $\alpha, \eta, \theta_0, \mathcal{D}_1 \dots \mathcal{D}_T$ **Output:**  $\theta_{\mathcal{T}}^*$ 

```

1:  $\theta \leftarrow \theta_0$ 
2: for  $t \leftarrow 1 \dots T$  do
3:   while Until Convergence do
4:      $(X, Y) \leftarrow$  Batch from  $D_t$ 
5:      $L \leftarrow \text{loss}(f(X), Y; \theta)$ 
6:     if  $t > 1$  then
7:        $\tilde{L} \leftarrow L + \alpha \sum_{t' < t} \sum_i F_i^{t'} (\theta_i - \theta_{t', i}^*)^2$ 
8:     else
9:        $\tilde{L} \leftarrow L$ 
10:    end if
11:     $g \leftarrow \nabla_{\theta} \tilde{L}$ 
12:     $\theta \leftarrow \theta - \eta g$ 
13:  end while
14:   $\theta_t^* \leftarrow \theta$ 
15:   $F_i^t \leftarrow \frac{1}{|D_t|} \sum_{(x, y) \sim D_t} \nabla_{\theta} f(x; \theta) \nabla_{\theta} f(x; \theta)^T$ 
16: end for

```

---

The derivation of *EWC* provided in Kirkpatrick *et al.* (2017) uses a two task case. Huszár (2017) extends the Laplace approximation in *EWC* to the multiple tasks leading to the following objective function:

$$\tilde{L}(\theta) = L_{\mathcal{T}}(\theta) + \alpha \sum_i \left[ \sum_{t < \mathcal{T}} \lambda^{(t)} F_i^t \right] (\theta_i - \theta_{\mathcal{T}-1, i}^*)^2. \quad (3.12)$$

This objective function depends only on the last set of optimal parameters  $\theta_{\mathcal{T}-1, i}^*$  and the sum of the previous Fisher matrices  $\sum_{t < \mathcal{T}} [\lambda^{(t)} F_i^t]$ . If the above modification is made, its memory complexity is constant in the number of tasks.

Approximating the Fisher information matrix using only the diagonal terms means that the interactions between the parameters are ignored. Ritter *et al.* (2018a) tries to obtain a better approximation by using the Kronecker product approximation of the Fisher information matrix, and it uses a block diagonal Fisher information matrix rather than a diagonal one. This translates to ignoring the interactions between

the parameters across the layers but taking into account the interactions within the same layer at the expense of having a higher computational complexity.

### 3.2.2 EWC++

Chaudhry *et al.* (2018) introduce *EWC++*, which is a more efficient version of *EWC*. *EWC++* uses the KL-divergence between the conditional probabilities  $p_\theta(y|x)$  and  $p_{\theta+\Delta\theta}(y|x)$  as a regularization loss. The conditional probability is represented by the neural network function  $f(x; \theta) := p_\theta(y|x)$ .

Here we follow the derivation for the  $D_{KL}(p_\theta||p_{\theta+\Delta\theta})$  provided in Chaudhry *et al.* (2018) that shows the relationship between the KL divergence and the Fisher information matrix.

Let's first start by defining the shorthand notations  $p_\theta(z) = p_\theta(y|x)$  and  $\mathbb{E}_z[\cdot] = \mathbb{E}_{x \sim \mathcal{D}, y \sim p_\theta(y|x)}[\cdot]$ , then we have:

$$D_{KL}(p_\theta(z)||p_{\theta+\Delta\theta}(z)) = \mathbb{E}_z [\log p_\theta(z) - \log p_{\theta+\Delta\theta}(z)] . \quad (3.13)$$

Using the second order Taylor expansion ( $z$  is omitted for brevity), we have:

$$\log p_{\theta+\Delta\theta} \approx \log p_\theta + \Delta\theta^\top \frac{\partial \log p_\theta}{\partial \theta} + \frac{1}{2} \Delta\theta^\top \frac{\partial^2 \log p_\theta}{\partial \theta^2} \Delta\theta . \quad (3.14)$$

By substituting this in Eq. (3.13), we get:

$$\begin{aligned} D_{KL}(p_\theta||p_{\theta+\Delta\theta}) &= \mathbb{E}_z [\log p_\theta] - \mathbb{E}_z [\log p_{\theta+\Delta\theta}] \\ &\approx \mathbb{E}_z [\log p_\theta] - \mathbb{E}_z [\log p_\theta] - \Delta\theta^\top \mathbb{E}_z \left[ \frac{\partial \log p_\theta}{\partial \theta} \right] \end{aligned} \quad (3.15)$$

$$\begin{aligned} &- \frac{1}{2} \Delta\theta^\top \mathbb{E}_z \left[ \frac{\partial^2 \log p_\theta}{\partial \theta^2} \right] \Delta\theta \\ &= -\Delta\theta^\top \mathbb{E}_z \left[ \frac{\partial \log p_\theta}{\partial \theta} \right] - \frac{1}{2} \Delta\theta^\top \mathbb{E}_z \left[ \frac{\partial^2 \log p_\theta}{\partial \theta^2} \right] \Delta\theta \end{aligned} \quad (3.16)$$



Here, the first term cancels out:

$$\begin{aligned}
\mathbb{E}_z \left[ \frac{\partial \log p_\theta(y|x)}{\partial \theta} \right] &= \mathbb{E}_{x \sim \mathcal{D}} \left[ \sum_y p_\theta(y|x) \frac{\partial \log p_\theta(y|x)}{\partial \theta} \right] \\
&= \mathbb{E}_{x \sim \mathcal{D}} \left[ \sum_y p_\theta(y|x) \frac{1}{p_\theta(y|x)} \frac{\partial p_\theta(y|x)}{\partial \theta} \right] \\
&= \mathbb{E}_{x \sim \mathcal{D}} \left[ \sum_y \frac{\partial p_\theta(y|x)}{\partial \theta} \right] \\
&= \mathbb{E}_{x \sim \mathcal{D}} \left[ \frac{\partial}{\partial \theta} \sum_y p_\theta(y|x) \right] = \mathbb{E}_{x \sim \mathcal{D}} [0] \\
\mathbb{E}_z \left[ \frac{\partial \log p_\theta(y|x)}{\partial \theta} \right] &= 0. \tag{3.17}
\end{aligned}$$

This means that Eq. (3.16) simplifies to:

$$D_{KL}(p_\theta || p_{\theta + \Delta\theta}) \approx -\frac{1}{2} \Delta\theta^\top \mathbb{E}_z \left[ \frac{\partial^2 \log p_\theta}{\partial \theta^2} \right] \Delta\theta = -\frac{1}{2} \Delta\theta^\top H \Delta\theta \approx \frac{1}{2} \Delta\theta^\top F \Delta\theta, \tag{3.18}$$

where  $F$  is the empirical Fisher information matrix and  $F = -H$  at the maximum likelihood estimate. As  $F$  is prohibitively expensive, the diagonal Fisher is used instead (which assumes that the parameters are independent), which gives:

$$D_{KL}(p_\theta || p_{\theta + \Delta\theta}) \approx \sum_i F_i \Delta\theta_i^2. \tag{3.19}$$

Chaudhry *et al.* (2018) proposes  $EWC++$  which uses the  $D_{KL}$  as the regularization term. After task  $\mathcal{T}$  is introduced, we have:

$$\tilde{L}(\theta) = L_{\mathcal{T}}(\theta) + \alpha D_{KL}(p_\theta || p_{\theta_{\mathcal{T}-1}^*}) \approx L_{\mathcal{T}} + \alpha \sum_i F_i (\theta_i - \theta_{\mathcal{T}-1,i}^*)^2. \tag{3.20}$$

It has to be noted that many of the approximations that were performed were based on the assumption that  $\Delta\theta$  is small. If the new  $\theta$  diverges away from the old  $\theta$ , these approximations become very imprecise, but that should not happen given that  $\theta$  is already constrained in the objective function to stay as close to the older  $\theta$  as possible.

We can see from Eq. (3.20) that  $EWC++$  is equivalent to  $EWC$  (Eq. (3.8)) when  $\mathcal{T}$  represents the second task, while they start to diverge

afterwards (Eq. (3.11)). *EWC++* is more efficient than *EWC* as it only needs to keep a single Fisher matrix and a single set of parameters  $\theta_{\mathcal{T}-1}$ , which gives it a constant memory complexity.

Another difference between *EWC* and *EWC++* is that in *EWC++*, the Fisher matrix is updated in an online fashion, and hence no extra pass over the dataset is needed after the task  $\mathcal{T}$  is done. After each minibatch, the diagonal Fisher matrix  $F$  is updated as follows:

$$F = \gamma F_{new} + (1 - \gamma) F_{old} \quad (3.21)$$

The pseudo code for *EWC++* is given in Algorithm 2.

---

**Algorithm 2** EWC++
 

---

**Input:**  $\alpha, \eta, \gamma, \theta_0, \mathcal{D}_1 \dots \mathcal{D}_T$

**Output:**  $\theta_{\mathcal{T}}^*$

```

1:  $\theta \leftarrow \theta_0$ 
2:  $F \leftarrow I$ 
3: for  $t \leftarrow 1 \dots \mathcal{T}$  do
4:   while Until Convergence do
5:      $(X, Y) \leftarrow$  Batch from  $D_t$ 
6:      $L \leftarrow \text{loss}(f(X), Y; \theta)$ 
7:      $F_{old} \leftarrow F$ 
8:      $F_{new} \leftarrow \frac{1}{|X|} \sum_{x \in X} \nabla_{\theta} f(x; \theta) \nabla_{\theta} f(x; \theta)^T$ 
9:      $F = \gamma F_{new} + (1 - \gamma) F_{old}$ 
10:    if  $t > 1$  then
11:       $\tilde{L} \leftarrow L + \alpha \sum_i F_i (\theta_i - \theta_{t-1,i}^*)^2$ 
12:    else
13:       $\tilde{L} \leftarrow L$ 
14:    end if
15:     $g \leftarrow \nabla_{\theta} \tilde{L}$ 
16:     $\theta \leftarrow \theta - \eta g$ 
17:  end while
18:   $\theta_t^* \leftarrow \theta$ 
19: end for

```

---

### 3.2.3 Synaptic Intelligence

Synaptic Intelligence (SI) (Zenke *et al.*, 2017) tries to measure the importance of the parameters by estimating how much each parameter affects the loss trajectory. In more concrete terms, the contribution of each parameter  $\theta_i$  to the loss during the current task  $\mathcal{T}$  is defined as:

$$\omega_i^{(\mathcal{T})} = \int_{t'_{\mathcal{T}-1}}^{t'_{\mathcal{T}}} \frac{\partial L_{\mathcal{T}}}{\partial \theta_i} \theta'_i(t') dt', \quad (3.22)$$

where  $t'$  is the time step and  $\theta'_i(t') dt'$  contributes to the parameter change from the initial point (at time  $t'_{\mathcal{T}-1}$ ) to the final point (at time  $t'_{\mathcal{T}}$ ). Since gradient descent uses discrete time steps to perform the updates, the effect of each parameter during task  $\mathcal{T}$  ( $\omega_i^{(\mathcal{T})}$ ) is, in practice, calculated as an online running sum of the product of the gradient with the parameter update  $\theta'_{\mathcal{T},i}(t')$ , where  $\omega_i^{(\mathcal{T})}$  is initialized to zero for each new task  $\mathcal{T}$ .

The parameter importance  $\Omega_i^{(\mathcal{T})}$  from Eq. (3.4) is calculated to be directly proportional to the contribution  $\omega_i^{(\mathcal{T})}$  of each parameter to the loss during the task  $\mathcal{T}$ , normalized with the square of final change  $\theta_i$  needed to make during  $\mathcal{T}$  to avoid large changes to important parameters (similar to Eq. (3.10)):

$$\Omega_i^{(\mathcal{T})} = \sum_{t < \mathcal{T}} \frac{\omega_i^{(t)}}{(\theta_{t,i}^* - \theta_{t-1,i}^*)^2 + \epsilon} \quad (3.23)$$

where  $\epsilon$  is the damping parameter used to bound the expression in case  $(\theta_{t,i}^* - \theta_{t-1,i}^*) \rightarrow 0$ .

### 3.2.4 Memory Aware Synapses (MAS)

We have seen in Section 3.2.1 (Elastic Weight Consolidation) that the importance of the parameter is inversely related to its uncertainty, for which the Fisher information matrix acts as a proxy. Hence a parameter with a higher precision is a more important parameter (from a probabilistic point of view as in Eq. (3.7)). Aljundi *et al.* (2018), on the other hand, try to estimate the importance of a parameter based on how sensitive the learned function  $F_{\mathcal{T}-1}$  is to a change in that

parameter. This removes the dependence on a labeled set to estimate the importance, and hence an unlabeled set can be used to estimate the parameter importance. Hence for a specific output class  $c$ , and a set of unlabeled dataset (or just the new task set)  $D$ , the sensitivity would be measured using the following equation:

$$\Omega_i = \frac{1}{|D|} \sum_{x \sim D} \frac{\partial F_c(x; \theta)}{\partial \theta_i}. \quad (3.24)$$

As an alternative to computing the importance per output/class, Aljundi *et al.* (2018) propose using the  $L_2$  norm of all the output nodes  $\|F(x; \theta)\|_2^2$  as a representative for all the outputs. Hence, the alternative equation would be (similar to Eq. (3.10)):

$$\Omega_i = \frac{1}{|D|} \sum_{x \sim D} \frac{\partial \|F(x; \theta)\|_2^2}{\partial \theta_i}. \quad (3.25)$$

Calculating the importance  $\Omega_i$  this way allows for decoupling the updates of the importance parameter  $\Omega$ , and the training on the task itself, since there is no dependence of  $\Omega$  on the task data. As mentioned earlier,  $D$  can be an unlabeled set of samples that are fixed across the tasks, but it also allows the use of an online stream of samples.

Benzing (2021) shows that although *SI* and *MAS* are motivated differently than *EWC*, as elaborated earlier, they both approximate the square root of the Fisher Information Matrix, which gives a unified picture for the three importance-based methods.

There have been some other recent advances in importance-based regularization techniques in lifelong learning. For example, building upon *MAS* (Aljundi *et al.*, 2018), Importance Driven Continual Learning approach (Özgün *et al.*, 2020) defines a parameter-specific learning rate such that the learning rate becomes a function of the parameter’s importance.

Jung *et al.* (2020) proposed Adaptive Group Sparsity based lifelong learning that introduced a loss function based on group-sparsity norms for parameter-wise importance regularization in neural networks.

### 3.3 Bayesian-Based Regularization

Bayesian-based regularization can be considered a special type of importance-based regularization. In some scenarios, we don't have access or are not allowed to store previously seen data due to privacy or security restrictions. In such scenarios, the lifelong learning algorithm's goal is to train a model at the current task using training data related to the current task without revisiting the training data from the previous task and reducing catastrophic forgetting through time. In this section, we revisit this scenario from a Bayesian inference perspective. The goal of the model for the first task is to predict a set of targets denoted as  $Y^{(t_1)}$ , where  $t_1$  is the task ID, in a supervised manner using parameters  $\theta$ , with a group of hyperparameters that the model uses to reach its highest performance. For a set of  $n$  i.i.d. samples used for training in the first task, the joint probability distribution of  $Y^{(t_1)}$  and model's parameters, given the hyperparameters used in the training procedure, can be formalized as follows:

$$p\left(Y^{(t_1)}, \theta \mid \alpha, X\right), \quad (3.26)$$

where  $X$  is the set of observation for task  $t_1$  and  $\alpha$  represents the hyperparameters. Computing the integral over  $\theta$  gives the desired marginal distribution  $\int p\left(Y^{(t_1)}, \theta \mid \alpha, X\right) d\theta = p\left(Y^{(t_1)} \mid \alpha, X\right)$ . By dividing and normalizing the joint distribution, we can also get the posterior distribution as  $p\left(\theta \mid Y^{(t_1)}, \alpha, X\right)$ . The model can be trained to predict both the posterior distribution and marginal distribution: (i) predicting the targets given by the best hyperparameters and (ii) having the best distribution of the model parameters given by targets and the set of observations. For lifelong learning, the posterior distribution can be obtained by multiplying the previous posterior by the likelihood of the dataset belonging to the new task and the regularization term can be interpreted as a prior.

We cannot compute the posterior distribution directly since we do not have any knowledge of the model parameters. However, we can expand the probability distribution using Bayesian inference and, from that, try to find the best distribution for model parameters. So,

according to the Bayesian inference for  $n$  i.i.d. training samples:

$$p\left(Y^{(t_1)}, \theta \mid \alpha, X\right) = \left[ \prod_{i=1}^n p\left(y_i^{(t_1)} \mid \theta, \alpha, x_i\right) \right] p(\theta \mid \alpha, X), \quad (3.27)$$

where  $(x_i, y_i)$  is a input and target pair from the observation set.

### 3.3.1 Approximate Inference in Lifelong Learning

In Eq. (3.27), calculating  $p(\theta \mid \alpha, X)$  is intractable in most cases i.e., it is not computationally possible to perform exact inference when the dimension of  $\theta$  is high. This motivates us to approximate the exact posterior distribution by another distribution that is computationally easier to handle. We can approximate  $p(\theta \mid \alpha, X)$  with a posterior distribution using prior knowledge that we have over  $\theta$  by assuming that it comes from a Gaussian distribution. With this assumption, we can get  $q^*(\theta \mid \alpha, X)$  as a posterior of the  $p(\theta \mid \alpha, X)$  using the Hidden Markov Model (HMM) or Gaussian processes approaches. Therefore, we can define the joint probability distribution for the second task as follows:

$$p\left(Y^{(t_2)}, Y^{(t_1)}, \theta \mid \alpha, X\right) = \left[ \prod_{i=n+1}^{n+m} p\left(y_i^{(t_2)} \mid \theta, \alpha, x_i\right) \right] p\left(Y^{(t_1)}, \theta \mid \alpha, X\right) \quad (3.28)$$

where  $n$  and  $m$  refer to number of samples in tasks  $t_1$  and  $t_2$  respectively. Substituting  $p\left(Y^{(t_1)}, \theta \mid \alpha, X\right)$  according to Eq. (3.27) we have:

$$\begin{aligned} p\left(Y^{(t_2)}, Y^{(t_1)}, \theta \mid \alpha, X\right) &= \left[ \prod_{i=n+1}^{n+m} p\left(y_i^{(t_2)} \mid \theta, \alpha, x_i\right) \right] \left[ \prod_{i=1}^n p\left(y_i^{(t_1)} \mid \theta, \alpha, x_i\right) \right] \\ &\quad \times p(\theta \mid \alpha, X) \\ &\approx \left[ \prod_{i=n+1}^{n+m} p\left(y_i^{(t_2)} \mid \theta, \alpha, x_i\right) \right] q_1^*(\theta, X) \\ &\approx q_2^*(\theta, X) \end{aligned} \quad (3.29)$$

The approximating distribution is usually chosen to be a product of several independent distributions, one for each parameter or a set of

similar parameters. Such methods have been used for solving various inference problems in machine learning.

There are several approaches for approximate inference, including Moment matching (Li *et al.*, 2015), variational KL minimization (Hershey *et al.*, 2007), Taylor expansion, Importance Sampling (Tokdar and Kass, 2010) and Laplace’s approximation (Friston *et al.*, 2007).

Approximate inference can be used to alleviate catastrophic forgetting in the lifelong learning setup. Proposed lifelong learning methods using approximate inference can be categorized as prior-focused methods. To this end, the main focus is on approximating the model’s parameters distribution. In a lifelong learning setup, we can approximate the model’s parameters recursively using the optimal parameters at previous tasks. This intuition and approach to find the best model parameters can show the importance of Approximate Inference to propose new methods in this setup. Both Taylor expansion and variational KL minimization can be used to alleviate catastrophic forgetting in lifelong learning methods. Next, we describe the proposed lifelong learning methods that employ these concepts in detail.

### 3.3.2 Variational KL Minimization

As we discussed in the EWC method in Section 3.2.1, the goal of regularization techniques is to tune the model parameters such that the parameters do not deviate from model parameters in the previous task. EWC alleviates the catastrophic forgetting by forcing model parameters to move around the last task parameters space considering the parameters’ importance using Fisher Information Matrix. To reach the same goal, Variational Continual Learning (VCL) method proposed an alternative way by using Approximate inference. To not deviate too much from previously learned parameters’ space, we can use the variational KL minimization to reduce the parameters’ distribution distance from what the model learned in the previous task (Nguyen *et al.*, 2017).

Before showing the advantage of using variational KL minimization to approximate the parameters’ posterior distribution, we review some basic divergence measures such as the Kullback-Leibler (KL) divergence

that is used to measure the closeness of the two distributions. It is defined as:

$$KL(q||p) = \mathbb{E}_q[\log(\frac{q(\theta)}{p(\theta)})], \quad (3.30)$$

where  $KL(q||p)$  indicates  $p$ 's divergence from  $q$ . Intuitively, when  $p(\theta)$  is large, but  $q(\theta)$  is small, there is a large divergence. When  $p(\theta)$  is small and  $q(\theta)$  is large, there will again be a large divergence, but not as large as the previous case.

VCL, as an influential method in prior-focused approaches, computes the posterior distribution of the parameters given the previous examples and keeps changing it over time. Computing the posterior distribution can be simplified using the mean-field approximation. VCL finds the new posterior for task  $t$  by that minimizes the KL-divergence with the old posterior at time step  $t - 1$  as follows:

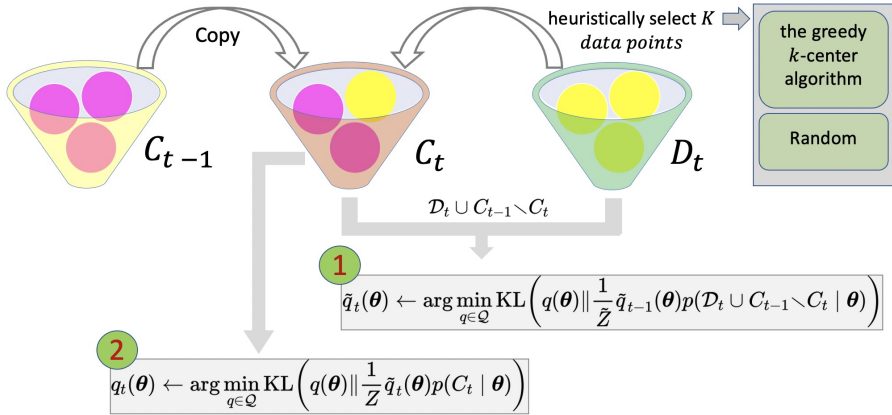
$$\tilde{q}_t(\theta) = \arg \min_{q \in Q} KL(q(\theta) || \frac{1}{Z_t} \tilde{q}_{t-1}(\theta) p(\mathcal{D}_t | \theta)) \quad (3.31)$$

where  $D_t$  is the training data at time  $t$ ,  $Z_t$  represents the intractable normalizing constant (Nguyen *et al.*, 2017). VCL predicts the targets for the test set inputs denoted as  $x^*$  as follow:

$$p(y^* | \mathbf{x}^*, \mathcal{D}_{1:t}) = \int q_t(\theta) p(y^* | \theta, \mathbf{x}^*) d\theta, \quad (3.32)$$

where  $\mathcal{D}_{1:t}$  represents the data from the beginning to the end of time  $t$ . To get the advantage of keeping some samples from the previous task in the replay, VCL proposes using a replay buffer called Coreset in the VCL-Coreset version. Figure 3.2 illustrates the VCL-Coreset algorithm. The VCL-Coreset observes the current task data denoted as  $D_t$ . It updates the coreset combining the information currently existing in the Coreset and  $D_t$  denoted as  $C_t$ . Then VCL updates the variational distribution for all samples in  $\mathcal{D}_t \cup C_{t-1} \setminus C_t$  as marked 1 in the Figure 3.2. VCL uses the sample in the  $C_t$  to compute the final variational distribution, which is used only for prediction and not propagation as marked by 2 in the figure 3.2. VCL uses the Eq. (3.32) to perform prediction at test time.





**Figure 3.2:** The summary of the Variational Continual Learning (VCL) algorithm.

To create a new Coreset  $C_t$  at time  $t$ , VCL selects new data points from the current task and a selection from the old coreset  $C_{t-1}$  as shown in figure 3.2. To select samples, any heuristic, including greedy approaches or simple random selection, can be used to select  $K$  data points from  $\mathcal{D}_t$  and added to  $C_{t-1}$ . It helps to have an unbiased coreset for computing the model parameters for the current task.

Following the VCL, Functional Regularisation for Continual Learning with Gaussian Processes (FRCL) (Titsias *et al.*, 2020) uses Gaussian process for a function family  $f = (f_1, \dots, f_n)$  such that functions sampled from  $\mathcal{N}(\mu, \Sigma)$  where  $\mu$  and  $\Sigma$  is defined by a mean function  $mean(x)$  and covariance function  $K(x, x')$  such that  $f(x) \sim \mathcal{GP}(mean(x), K(x, x'))$ . In FRCL each function  $f_i$  defined as follow:

$$f_i(x; w_i) \equiv f_i(x; w_i, \theta) = w_i^\top \phi(x; \theta) \quad (3.33)$$

$$\text{and } w_i \sim \mathcal{N}(w_i \mid 0, \sigma_w^2 I), \quad (3.34)$$

where  $\Omega_i$  is the task-specific weights and  $\phi(x; \theta)$  represents the shared feature vector. FRCL should maximize  $\mathcal{F}(\theta, q(w_k))$  as a learning ob-

jective that is computed as follow:

$$\begin{aligned} \mathcal{F}(\theta, q(w_k)) &= \sum_{j=1}^{N_k} \mathbb{E}_{q(w_k)} \left[ \log p(y_{k,j} | w_k^\top \phi(x_{k,j}; \theta)) \right] \\ &\quad - \text{KL}(q(w_k) \| p(w_k)) \\ &\quad - \sum_{i=1}^{k-1} \text{KL}(q(u_i) \| p_\theta(u_i)) \end{aligned}$$

where,

$$\begin{aligned} f_k(x; w_k) &= w_k^\top \phi(x; \theta), \\ w_k &\sim \mathcal{N}(0, \sigma_w^2 I) \\ q(w_k) &= \mathcal{N}(w_k | \mu_{w_k}, \Sigma_{w_k}) \end{aligned}$$

The  $-\sum_{i=1}^{k-1} \text{KL}(q(u_i) \| p_\theta(u_i))$  term is also considered as the regularisation term ( $R_{\mathcal{T}}$  in Eq. (3.2)) that is computed for the previous tasks (Titsias *et al.*, 2020). FRCL uses the  $KL$  term to distinguish the task boundaries such that if  $\text{KL} \gg 0$  shows the task shift and  $\text{KL} \approx 0$  shows that model is still in the same task.

Recently, Uncertainty-regularized Continual Learning proposed by (Ahn *et al.*, 2019) builds on a Bayesian learning framework with variational inference with the notion of node-wise uncertainty. The authors perform an interpretation of the closed-form of the KL-divergence term for the Gaussian mean-field approximation and the Bayesian neural network pruning that reduces the number of additional parameters for implementing per-weight regularization. On the other hand, Uncertainty-guided Continual Bayesian Neural Networks (Ebrahimi *et al.*, 2019) introduced a learning rate that adapts according to the uncertainty defined in the probability distribution of the weights in networks and retains task performance after pruning weights by saving binary masks per task. Kumar *et al.* (2021) applies variational Bayesian-based regularization for both discriminative and generative settings by learning priors from previous tasks.

### 3.4 Distillation-based Regularization

Distillation-based regularization is mainly based on the following premises: if the network has access to the samples of task 1, and was forced to produce the same output on these samples while training on task 2, then the performance on task 1 would be preserved and no catastrophic forgetting would take place. However, the samples of task 1, evidently, are not there anymore, but since images lie on a low dimensional manifold (Pless and Souvenir, 2009), images of the new task may provide some sort of sampling of the images of task 1, and hence, preserving the network output on these images for the classes of task 1 may keep the performance on task 1 from degrading, depending on how similar the images are between the two tasks. As this mechanism is similar to knowledge distillation (Gou *et al.*, 2021), where the teacher network is the network trained on task 1 and the student network is the network being trained on task 2, we shall call it distillation-based regularization.

#### 3.4.1 Learning Without Forgetting

The distillation based regularization in the context of lifelong learning was introduced by Learning Without Forgetting (LWF) (Li and Hoiem, 2017). A copy of the model that was trained on the last task ( $f_{\mathcal{T}-1}$ ) is saved. When a new task  $\mathcal{T}$  is introduced, the output of  $f_{\mathcal{T}-1}$  is used as a soft target for the new model  $f_{\mathcal{T}}$  to imitate. This happens for the classes that are shared between the two models (the classes that belong to the previous tasks). This idea is inspired from knowledge distillation (Hinton *et al.*, 2015), where the model trained on the previous task  $f_{\mathcal{T}-1}$  is the teacher model, and the model being trained on the current task  $f_{\mathcal{T}}$  is the student model. Since the student model is only trained on the current task data, LWF is making the assumption that the samples of the current task might provide a poor sampling for the older tasks. Hence, if there is visual similarity between the previous tasks and the new tasks, this mechanism can help in alleviating catastrophic forgetting. iCaRL (Rebuffi *et al.*, 2017) extends LWF to the case when there is a replay buffer that provides samples for the older tasks (Chapter 4).

In more concrete terms, the objective function that LWF tries to

minimize when training on task  $\mathcal{T}$  is:

$$\tilde{L}(\theta) = - \underbrace{\sum_{c \in \mathcal{C}(\mathcal{T})} \delta_{c=y} \log f_c(x; \theta)}_{L_{\mathcal{T}}(\theta) \text{ in Eq. (3.2)}} - \underbrace{\sum_{c \in \mathcal{Y}(\mathcal{T}-1)} f_c(x; \theta_{\mathcal{T}-1}^*) \log f_c(x; \theta)}_{R_{\mathcal{T}} \text{ in Eq. (3.2)}} \quad (3.35)$$

Along with the cross-entropy loss for the new task (first part), the knowledge distillation loss (second part) is incorporated to impose output stability of old tasks with new data. A temperature term might be multiplied to the distillation loss in order to control how sensitive is the distillation loss to the difference between  $f_c(x; \theta_{\mathcal{T}-1}^*)$  and  $f_c(x; \theta)$ .

### 3.4.2 Learning without Memorization

Learning without Memorization (LWM) (Dhar *et al.*, 2019) claims that imitating the output of the previous model is not enough, but also the model has to remember where to look at; what are the regions in the image that it used to look at before. The intuition is that the new model  $f_{\mathcal{T}}$  has no extra information about the older classes than the old model  $f_{\mathcal{T}-1}$ , and hence it makes no sense to look elsewhere when evaluating the older classes. In addition to the distillation on the output of the teacher model, it applies a distillation loss on the attention maps of the teacher model for the classes that belong to the old tasks as following:

$$\tilde{L}(\theta) = L_{\mathcal{T}}(\theta) + \beta L_D(\theta) + \gamma L_{AD}(\theta) \quad (3.36)$$

where  $\beta$  and  $\gamma$  are regularization parameters,  $L_D(\theta)$  is the distillation loss as used in LWF (3.4.1) and  $L_{AD}(\theta)$  is the attention distillation loss that is defined as the sum of element wise  $L_1$  difference of the normalized, vectorized attention map (generated using Grad-CAM (Selvaraju *et al.*, 2017)). Therefore, the attention maps represent the regions in the image which resemble the base classes.

Deep Model Consolidation (DMC) (Zhang *et al.*, n.d.) leverages the unlabeled auxiliary data instead of old training data to ensure the student model absorbs the knowledge in an unbiased way. Essentially, it learns a new network for each new task and then trains a new model on the outputs of the old and new networks on some unlabeled data to promote symmetric knowledge transfer. The premise is that if natural

images lie on a low-dimensional manifold, then the unlabeled data from a similar domain will provide some representation for the datasets of the learned tasks.

Recent knowledge distillation-based regularization methods in lifelong learning includes Batch-level Distillation (BLD) (Fini *et al.*, 2020) that adopts a dynamic weighting strategy while minimizing the memory overhead. Similarly, Zhong *et al.* (2021) proposed a discriminative distillation approach by adding an expert classifier whose knowledge is distilled to the new classifier to discriminate features between confusing classes in lifelong learning. Douillard *et al.* (2021) proposed a multi-scale pooling distillation approach to preserve long and short-range spatial relationships at the feature level. To avoid catastrophic forgetting of the old classes, they perform an entropy-based pseudo-labeling of the background for the classes predicted by the old model.

### 3.5 Optimization Trajectory based Regularization

Another direction in the lifelong learning research is to look at the problem from the perspective of the optimization. Mirzadeh *et al.* (2020b) has shown that the geometric nature of the local minima reached by the learning algorithm affects how much the model is affected by catastrophic forgetting. Let  $w_i^*$  be the minima obtained after sequential training on the  $i^{th}$  task and  $L_j(w_i)$  as the loss of  $j^{th}$  task with parameters  $w_i$ . Then forgetting of the first task  $F_1$  after training on the second task is defined as:

$$F_1 := L_1(w_2^*) - L_1(w_1^*) \quad (3.37)$$

According to the second-order Taylor expansion of  $L_1(w_2^*)$  around  $w_1^*$ :

$$L_1(w_2^*) \approx L_1(w_1^*) + \frac{1}{2}(w_2^* - w_1^*)^\top \nabla^2 L_1(w_1^*)(w_2^* - w_1^*) \quad (3.38)$$

By using the above approximation to  $L_1(w_2^*)$ , Mirzadeh *et al.* (2020b) derived an upper bound to  $F_1$  in terms of the maximum eigen value ( $\lambda_1^{max}$ ) of  $\nabla^2 L_1(w_1^*)$ :

$$\begin{aligned} F_1 &\approx \frac{1}{2}(w_2^* - w_1^*)^\top \nabla^2 L_1(w_1^*)(w_2^* - w_1^*) \\ &\leq \frac{1}{2}\lambda_1^{max} \|w_2^* - w_1^*\|_2^2, \end{aligned} \quad (3.39)$$

According to the bound in Eq. (3.39), smaller  $\lambda_1^{max}$  means lesser forgetting.  $\lambda_1^{max}$  has been used to characterize the width of the local minima – small values correspond to flat minima, and large values correspond to sharp/ narrow minima (Hochreiter and Schmidhuber, 1997a). Based upon the above analysis and empirical results on existing benchmarks, Mirzadeh *et al.* (2020b) conclude that flatter minima lead to lesser forgetting. Building on this result, Mehta *et al.* (2021) has shown that models initialized with pre-trained weights undergo lesser forgetting than random weights as pre-trained models have an inductive bias towards flat task minima.

### 3.5.1 Sharpness Aware Minimization (SAM)

To promote flat minima during lifelong learning, Mirzadeh *et al.* (2020b) suggests modifying training dynamics by varying hyper-parameters like batch size, learning rate, and dropout regularization. It is well known that these hyper-parameters influence the optimization trajectory and loss curvature around minima (Xie *et al.*, 2021), also the variance of the gradients (Jastrzebski *et al.*, 2020), implicitly leading to flatter minima for certain values. However, searching for appropriate hyper-parameters for lifelong learning is ill-defined as one does not know task sequence a priori. To address these limitations, Mehta *et al.* (2021) proposes to explicitly optimize for the loss sharpness (alternatively flatter minima) during lifelong learning. Specifically, they employ a Sharpness-Aware Minimization (SAM) procedure (Foret *et al.*, 2021) to simultaneously minimize task loss value and loss sharpness.

SAM searches for parameters that lie in neighborhoods with uniformly low loss regions by minimizing the following loss sharpness (for model  $f$  with parameters  $w$ ):

$$\max_{\|\epsilon\|_2 \leq \rho} f(w + \epsilon) - f(w), \quad (3.40)$$

where the maximization region is defined to be  $\ell^2$  ball with radius  $\rho$  around  $w$ . Formally, the SAM procedure comprises solving the following

minimax optimization problem:

$$\begin{aligned} & \min_w \underbrace{f(w)}_{\text{task loss}} + \underbrace{\max_{\|\epsilon\|_2 \leq \rho} f(w + \epsilon) - f(w)}_{\text{loss sharpness}} + \underbrace{\lambda \|w\|_2^2}_{\text{L2 regularization}} \\ & \min_w \max_{\|\epsilon\|_2 \leq \rho} f(w + \epsilon) + \lambda \|w\|_2^2 \end{aligned} \quad (3.41)$$

Let  $\hat{\epsilon}(w)$  denotes the solution to the inner maximization problem in Eq. (3.41). By using first-order Taylor expansion of  $f(w + \epsilon)$  w.r.t  $\epsilon$  around 0 and solving for dual norm problem, Foret *et al.* (2021) derives  $\hat{\epsilon}(w)$  to be:

$$\hat{\epsilon}(w) = \rho \frac{\nabla_w f(w)}{\|\nabla_w f(w)\|_2} \quad (3.42)$$

By using the value of  $\hat{\epsilon}(w)$  from Eq. (3.42), the gradient for the minimax problem in Eq. (3.41) is approximated as:

$$\begin{aligned} \nabla_w \max_{\|\epsilon\|_2 \leq \rho} f(w + \epsilon) & \approx \nabla_w f(w)|_{w+\hat{\epsilon}(w)} + \frac{\partial \hat{\epsilon}(w)}{\partial w} \nabla_w f(w)|_{w+\hat{\epsilon}(w)} \\ & \approx \nabla_w f(w)|_{w+\hat{\epsilon}(w)} \quad (\text{dropping the second order gradient term}) \end{aligned} \quad (3.43)$$

### 3.5.2 Orthogonal Gradient Descent (OGD)

OGD proposed by Farajtabar *et al.* (2020) works on the optimization trajectory by projecting the gradients of the new task  $\mathcal{T}$  in a direction that is still useful for learning the task  $\mathcal{T}$ , but that does not change the predictions on the older tasks  $t < \mathcal{T}$ . In other words, it tries to maintain a space of the gradient directions of the predictions in the previous tasks, and project the gradients in the new task in a direction perpendicular to that space (Figure 3.3).

In more concrete terms, given a task  $A$  which contains  $c$  classes and a dataset  $D_A$  containing  $n_A$  samples, we would have  $n_A \times c$  gradient directions  $\nabla_{\theta} f_j(x; \theta) \forall j \leq c$ , where  $f_j$  is the model output for class  $j$ . During training on task  $B$  when obtaining the gradient  $g$ , *OGD* tries to find the gradient direction  $\tilde{g}$  such that:

$$\tilde{g} \perp \nabla_{\theta} f_j(x; \theta) \quad \forall \quad j \leq c, x \in D_A. \quad (3.44)$$

In practice, instead of keeping  $\nabla_{\theta} f_j$  for all classes, the average of the labels can be kept, or only the ground truth class can be kept as well (which is what is mainly adopted in Farajtabar *et al.* (2020)). Moreover, since obtaining the exact  $\nabla_{\theta} f(x; \theta)$  for any  $\theta$  requires having the whole dataset  $D_A$ , only the gradients at the optimum  $\theta_A^*$  are kept, assuming that the optimization of the subsequent tasks will let the parameters  $\theta$  stay in the vicinity of  $\theta_A^*$ . Finally, not all the gradients are kept for the whole  $D_A$ , but rather a subset of these gradients that is chosen randomly.

As mentioned before, when training on task  $\mathcal{T}$ , the gradient  $\tilde{g}$  should be perpendicular to the space of the gradients of the ground truth logits for the previous tasks  $t < \mathcal{T}$ . This space is defined as:

$$S = \text{span}\{\nabla f_k(x, \theta_t^*) | (x, y) \in D_t, t < \mathcal{T}, y_k = 1\}. \quad (3.45)$$

The orthogonal basis for  $S$  are obtained using the Gram-Schmidt procedure, where they are computed iteratively as follows:

$$\begin{aligned} v_1 &= \nabla f_{k_{1,1}}(x_{1,1}; \theta_1^*) \\ v_i &= \nabla f_{k_{t,l}}(x_{t,l}; \theta_t^*) - \sum_{j < i} \text{proj}_{v_j}(\nabla f_{k_{t,l}}(x_{t,l}; \theta_t^*)) \\ v_n &= \nabla f_{k_{\mathcal{T}-1, n_{\mathcal{T}-1}}}(x_{\mathcal{T}-1, n_{\mathcal{T}-1}}; \theta_{\mathcal{T}-1}^*) - \sum_{j < n} \text{proj}_{v_j}(f_{k_{\mathcal{T}-1, n_{\mathcal{T}-1}}}(x_{\mathcal{T}-1, n_{\mathcal{T}-1}}; \theta_{\mathcal{T}-1}^*)) \end{aligned} \quad (3.46)$$

Where  $n_t = |D_t|$ ,  $n = \sum_{t < \mathcal{T}} n_t$ .  $k_{t,l}$  represents the ground truth index for sample  $l$  in  $D_t$  such that  $y_{t, l_{k_{t,l}}} = 1$ . Finally,  $\text{proj}_v(u) = \frac{\langle u, v \rangle}{\langle v, v \rangle} v$ . After obtaining the space  $S$ , the gradient  $\tilde{g}$  is obtained using:

$$\tilde{g} = g - \sum_{v \in S} \text{proj}_v(g) \quad (3.47)$$

where it is proven in Farajtabar *et al.* (2020) that  $\tilde{g}$  is still a descent direction for task  $\mathcal{T}$ , where there exists a non zero learning rate  $\eta$ , such that taking a step in the direction  $\eta \tilde{g}$  will reduce the loss on task  $\mathcal{T}$ .



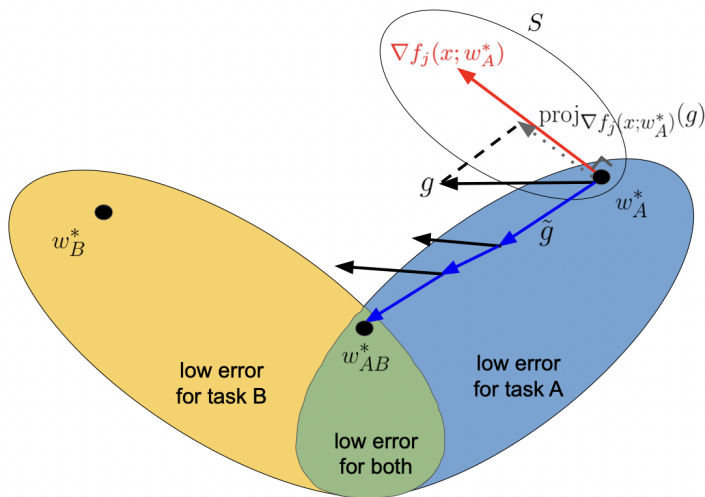


Figure 3.3: Orthogonal Gradient Descent

---

**Algorithm 3** Orthogonal Gradient Descent

---

**Input:**  $\theta_0, \mathcal{D}_1 \dots \mathcal{D}_T$

**Output:**  $\theta_T^*$

- 1:  $S \leftarrow \{\}$
  - 2:  $\theta \leftarrow \theta_0$
  - 3: **for**  $t \leftarrow 1 \dots T$  **do**
  - 4:   **while** Until Convergence **do**
  - 5:      $X \leftarrow$  Batch from  $D_t$
  - 6:      $g \leftarrow \nabla_{\theta}$  Gradient for  $X$  at  $\theta$
  - 7:      $\tilde{g} \leftarrow g - \sum_{v \in S} \text{proj}_v(g)$
  - 8:      $\theta \leftarrow \theta - \eta \tilde{g}$
  - 9:   **end while**
  - 10:  $\theta_t^* \leftarrow \theta$
  - 11: **for**  $(x, y)$  in  $D_t$  **do**
  - 12:    $v \leftarrow \nabla f(x; \theta_t^*) - \sum_{v \in S} \text{proj}_v(\nabla f(x; \theta_t^*))$
  - 13:    $S \leftarrow S \cup v$
  - 14: **end for**
  - 15: **end for**
-

### 3.5.3 Task-based Accumulated Gradients (TAG)

TAG proposed by Malviya *et al.* (2021) uses an adaptive learning rate based on the relatedness between tasks in the incremental task setup. TAG is applied mainly to the *RMSProp* optimization method, but Malviya *et al.* (2021) offers other versions for how TAG can be applied to other adaptive optimization methods as well.

*RMSProp* update rule works as follows:

$$V_n = \beta V_{n-1} + (1 - \beta)g_n^2 \quad (3.48)$$

$$\theta_{n+1} = \theta_n - \frac{\eta}{\sqrt{V_n} + \epsilon} g_n \quad (3.49)$$

Where  $n$  represents the update step,  $V_n$  is the moving average of the square of the gradients,  $\eta$  is the original learning rate, and  $\epsilon$  is a very small value to avoid degenerate cases. Although adaptive optimization methods tend to perform well in the normal supervised learning setup, they tend to perform poorly in the lifelong learning setup (Mirzadeh *et al.*, 2020b).

TAG tries to adapt *RMSProp* in a way that is more aligned with the lifelong learning setup. First, it captures the first moment of the gradient for each task  $t$ :

$$M_n^{(t)} = \gamma M_{n-1}^{(t)} + (1 - \gamma)g_n \quad (3.50)$$

This moment acts as a proxy for the adaptation trajectory that the model followed for task  $t$ , and hence the correlation  $\alpha_n(\mathcal{T}, t)$  between the different tasks  $t$  and  $\mathcal{T}$  can be measured by using this moment as follows:

$$\alpha_n(\mathcal{T}, t) = \exp\left(-b \frac{M_n^{(\mathcal{T})T} M_N^{(t)}}{|M_n^{(\mathcal{T})}| |M_N^{(t)}|}\right) \quad (3.51)$$

where  $M_n^{(\mathcal{T})}$  is the moment for the current task  $T$  after the  $n$ -th step, and  $M_N^{(t)}$  is the final moment computed on a previous task  $t$ .  $b$  is a hyperparameter.

They argue that if a task  $t$  is correlated with a previous task  $\mathcal{T}$ , the learning rate in the parameter update step would be higher to encourage the transfer of knowledge between task  $t$  and  $\mathcal{T}$ . Whereas if the current task  $t$  is uncorrelated or negatively correlated to a previous task  $\tau$ , the

new updates over parameters may cause catastrophic forgetting and hence the learning rate should adapt to lessen the effects of the new updates. By using  $\alpha_n(\mathcal{T}, t)$  as a proxy for the correlation between tasks, *TAG* modifies Eq. (3.48) to take this correlation into account for any current task  $\mathcal{T} > 1$  as follows:

$$\theta_{\mathcal{T},n+1} = \theta_{\mathcal{T},n} - \frac{\eta}{\sqrt{\alpha_n(\mathcal{T}, \mathcal{T})V_n^{(\mathcal{T})} + \sum_{t=1}^{\mathcal{T}-1} \alpha_n(\mathcal{T}, t)V_N^{(t)} + \epsilon}} g_n \quad (3.52)$$

Using the update rule this way ensures that the second moment of the gradients from the previous tasks is taken into account when adapting the learning rate, and that the more similar tasks have a stronger effect than the less similar tasks. With the exponential term,  $\alpha_n(t, \tau)$  will attain a higher value for uncorrelated tasks and will minimize the new updates (hence prevent forgetting).

### 3.6 Summary

To overcome catastrophic forgetting, the regularization-based methods presented in this chapter add a penalty term in the objective function, to constrain the drastic changes in the model parameters when data from a new task arrives. Based on the motivation for defining the penalty terms and storing the past knowledge, the regularization-based methods are further categorized.

The first type of approach involves quantifying the importance of each parameter with respect to the previous tasks and using this knowledge to control the new changes in the parameters. These approaches require storing the model parameters before it started learning the new task. We present several proposed methods that compute the importance-term in different ways.

Another closely related methods apply concepts of Bayesian inference in lifelong learning. These methods approximate the model's parameters distribution using the optimal parameters at previous tasks. In particular, we expand upon the two prominent approximate inference techniques employed to alleviate catastrophic forgetting: 1) Taylor expansion and 2) Variational KL minimization.

The importance-based approach and the Bayesian-based approach

rely on the prior knowledge based on model parameters. The third type of approach presented in this chapter is data-focused since it involves knowledge distillation to preserve knowledge by imposing output stability of past tasks with new data.

The final type of approach utilizes the optimization trajectories to impose hard constraints and apply learning rates that adapt for each task in lifelong learning. These approaches essentially require storing knowledge in the form of gradients computed during the parameter updates.

Since the regularization-based methods generally require storing model parameters from previous tasks, they are computationally expensive and often depend on the choice of prior. Moreover, when the model needs to adapt to a large number of tasks, the interference between task-based knowledge is inevitable with fixed model capacity. In the next chapter, we explore memory-based methods that do not require storing the model and comprise several state-of-the-art techniques in lifelong learning with fixed model capacity. We shall discuss methods with dynamic model capacity in the chapter after that.

# 4

---

## Memory-based Approaches

---

The vanilla approach for lifelong learning is to fine-tune the model parameters on a new task ( $t$ ) starting from the previous task parameters ( $\theta_{t-1}$ ). Chapter 3 focuses on parameter-based regularization approaches, which prevent the model parameters from deviating too far from their initialization while optimizing the current task loss. On the other hand, in this chapter, we discuss data-based regularization approaches that hope to induce a similar behavior as parameter-based approaches. Specifically, we focus on memory-based approaches for lifelong learning, and the main feature of such approaches is an *episodic memory*,  $\mathcal{M}_t$ . Episodic memory retains a subset of the observed examples from task  $t$ , and memory-based approaches use it to regularize the learning of the future tasks to alleviate forgetting of previous tasks.

Classical cognitive science studies view human memory as a single system, i.e., “memory is memory.” However, recent studies show that human memory consists of several components, each performing varied functionalities and operating under different principles. In one such study, Tulving (1985) proposes a ternary classification scheme of memory constituting procedural, semantic, and episodic memories. These three memories form a hierarchical arrangement - procedural

memory at the lowest level, followed by specialized semantic memory and episodic memory at the top level. Procedural memory helps deal stimulus patterns with response chains; semantic memory enables humans to construct mental models of the world (based on the capability of internally representing states of the world); episodic memory handles acquisition and retention of individual experiences and allows revisiting (replaying) them again. In another work, McClelland *et al.* (1995) proposes a theory of Complementary Learning Systems (CLS) which advocates that humans rely on episodic memory to store past experiences and conduct experience rehearsal to retain previously learned knowledge. Specifically, there are two complementary systems: one that allows for the gradual accumulation of knowledge and another that allows rapid adaptation to individual experiences.

Motivated by the above studies, a plethora of works (Mitchell *et al.*, 2018; Chen *et al.*, 2015c; Lopez-Paz and Ranzato, 2017b; Chaudhry *et al.*, 2019a; Sprechmann *et al.*, 2018; Masson d’Autume *et al.*, 2019; Wang *et al.*, 2020b; Riemer *et al.*, 2019; Guo *et al.*, 2020) employ memory modules for lifelong learning, particularly to alleviate the catastrophic forgetting phenomena. These works differ along multiple dimensions:

- What type of memory system is used? *Episodic memory* retains a subset of the observed examples for replay (Lopez-Paz and Ranzato, 2017b; Masson d’Autume *et al.*, 2019), *Semantic memory* retains the structured knowledge (Mitchell *et al.*, 2018; Chen *et al.*, 2015c; Schwarz *et al.*, 2018), *Generative memory* learns the parametric model of the data and reconstructs past task examples for replay (Shin *et al.*, 2017; Sun *et al.*, 2020).
- How is memory employed? *Explicit* constraints on the current task gradients (Lopez-Paz and Ranzato, 2017b; Chaudhry *et al.*, 2019a; Guo *et al.*, 2020), *Implicit* constraints on the current task gradients (Chaudhry *et al.*, 2019b; Riemer *et al.*, 2019; Sprechmann *et al.*, 2018).
- How is memory populated? *Random* examples (Chaudhry *et al.*, 2019b), *Uncertain* examples (Aljundi *et al.*, 2019d), *Forgettable* examples (Wang *et al.*, 2020b).

## 4.1 A Unified View of Episodic Memory for Lifelong Learning

Memory-based approaches use episodic memory ( $\mathcal{M} = \cup_{k < t} \mathcal{M}_k$ ) to replay the examples from previous tasks while updating the model with the current task  $t$ . While several methods have been developed, here we abstract away from their specific implementations and instead focus on a unified view of episodic memory-based approaches for lifelong learning.

**Problem Definition.** We consider a setup with continuum of task data:

$(x_1, t_1, y_1), \dots, (x_i, d_i, y_i), \dots, (x_n, t_n, y_n)$ . Each triplet  $(x_i, d_i, y_i)$  consists of a task descriptor  $d_i \in \mathcal{T}$ , input data  $x_i \in \mathcal{D}_{d_i}$  and target labels  $y_i \in \mathcal{Y}_{d_i}$ . Here we assume that an explicit task descriptor  $d_i$  is available. Further, we assume that the continuum is locally i.i.d., i.e., triplet  $(x_i, d_i, y_i)$  satisfies  $(x_i, y_i) \stackrel{iid}{\sim} \mathcal{P}_{d_i}(X, Y)$ . Overall, the goal is to learn a predictor  $f_\theta : \mathcal{X} \times \mathcal{T} \rightarrow \mathcal{Y}$  such as a neural network, parameterized by  $\theta \in \mathbb{R}^P$ , to minimize the average expected risk of all  $T$  tasks:

$$R(f_\theta) := \frac{1}{T} \sum_{t=1}^T \mathbb{E}_{x,y \sim P_t} [\ell(f(x, t; \theta), y)], \quad (4.1)$$

with  $\ell(\cdot, \cdot)$  being the specific task loss. While the average risk is commonly evaluated after the model has seen all tasks, one can also evaluate test pairs  $(x, t)$  from previously observed tasks at different stages to demonstrate the model's training behavior, and evaluate its robustness against catastrophic forgetting in terms of backward and forward transfer. While different methods have been developed to optimize Eq. (4.1), in this chapter we focus on memory based approaches for lifelong learning. The main feature of these approaches is an *episodic memory*,  $\mathcal{M}_t$ , which retains a subset of the observed examples from each task  $t$ .

**Algorithm 4** Unified View of Episodic Rehearsal for Lifelong Learning**Input:**  $\theta_0, \mathcal{M}, \mathcal{D}_1 \dots \mathcal{D}_T$ **Output:**  $\theta_T, \mathcal{M}$ 


---

```

1:  $\mathcal{M} \leftarrow \{\}$ 
2: for  $t \leftarrow 1 \dots T$  do
3:    $\theta_t \leftarrow \theta_{t-1}$ 
4:   for  $k \leftarrow 1 \dots |\mathcal{D}_t|$  do
5:      $b_t^k \leftarrow \mathcal{D}_t[k]$  {Sample mini-batch for current task}
6:      $r_t^k \leftarrow \{\}$ 
7:     if  $\mathcal{M} \neq \{\}$  then
8:        $r_t^k \leftarrow \text{MEMREAD}(\mathcal{M})$  {Sample replay examples from the
          episodic memory}
9:       ER: Set  $\alpha_1(\theta_t^k) = 1$  and  $\alpha_2(\theta_t^k) = 1$ 
10:      GEM: Set  $\alpha_1(\theta_t^k)$  and  $\alpha_2(\theta_t^k)$  based on Eq. (4.9)
11:      A-GEM: Set  $\alpha_1(\theta_t^k)$  and  $\alpha_2(\theta_t^k)$  based on Eq. (4.13)
12:      MEGA-I: Set  $\alpha_1(\theta_t^k)$  and  $\alpha_2(\theta_t^k)$  based on Eq. (4.14)
13:     else
14:       Set  $\alpha_1(\theta_t^k) = 1$  and  $\alpha_2(\theta_t^k) = 0$ 
15:     end if
16:      $\theta_t^{k+1} \leftarrow \text{UPDATE}(\theta_t^k, b_t^k, r_t^k)$  {Based on Eq. (4.5)}
17:      $\mathcal{M}_t \leftarrow \text{MEMWRITE}(\mathcal{M}, b_t^k)$ 
18:   end for
19:    $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_t$ 
20: end for

```

---

Formally, given a task  $t$ ,  $b_t$  denotes a mini-batch sampled from  $\mathcal{D}_t$ , Eq. (4.2) defines the task loss on  $b_t$  and Eq. (4.3) defines the replay loss on  $\mathcal{M}_t$ .

$$L_{TASK}(\theta; b_t) = \frac{1}{|b_t|} \sum_{(x,t,y) \in b_t} \ell(f(x, t; \theta), y), \quad (4.2)$$

$$L_{REP}(\theta; \mathcal{M}_t) = \frac{1}{|\mathcal{M}_t|} \sum_{(x,t,y) \in \mathcal{M}_t} \ell(f(x, t; \theta), y). \quad (4.3)$$

Algorithm 4 outlines the overall training procedure corresponding to a unified approach. There are three main routines in Algorithm 4 -



*MEMREAD*, *UPDATE* and *MEMWRITE* and most of the memory-based approaches differ in terms of a specific implementation of these. *MEMREAD* routine implements a strategy to sample examples from the memory while updating the model with current task gradients. *MEMWRITE* routine implements a strategy to select a subset of the observed examples to write to the memory for replay. *UPDATE* routine implements how two objectives - task loss Eq. (4.2) and replay loss Eq. (4.3) are combined to alleviate forgetting and enable backward/forward transfer. In this section, we discuss different realizations of the *UPDATE* routine, and Section 4.3 discusses the rest of the two routines in detail.

**UPDATE.** Let  $\theta_t^k$  denote the model parameters when training on  $k$ -th mini-batch of the task  $t$ . Under the unified view, the joint optimization problem covering the task loss and replay loss is defined as follows:

$$\min_{\theta} \alpha_1(\theta_t^k) L_{TASK}(\theta) + \alpha_2(\theta_t^k) L_{REP}(\theta). \quad (4.4)$$

Using stochastic gradient descent method to solve Eq. (4.4), one-step gradient descent update for model parameters starting with  $\theta_t^k$  is defined as follows:

$$\theta_t^{k+1} \leftarrow \theta_t^k - \eta(\alpha_1(\theta_t^k) \nabla_{\theta} L_{TASK}(\theta) + \alpha_2(\theta_t^k) \nabla_{\theta} L_{REP}(\theta)), \quad (4.5)$$

where  $\alpha_1(\theta), \alpha_2(\theta)$  are real-valued functions controlling the relative importance of  $L_{TASK}(\theta)$  and  $L_{REP}(\theta)$  in each mini-batch. Now we deep-dive into existing replay-based methods and see how they all are different manifestations of the update Eq. (4.5).

#### 4.1.1 Experience Replay (ER)

The most basic type of update strategy is to replay examples,  $M_k$ , for each task  $k$  learned so far while learning the current task  $t$ . Under the unified view, for the update Eq. (4.5) we have  $\alpha_1(\theta_t^k) = \alpha_2(\theta_t^k) = 1$  in Eq. (4.5). Chaudhry *et al.* (2019b) shows that as simple as this strategy seems, it performs exceedingly well compared to other more sophisticated algorithms.

### 4.1.2 Gradient Episodic Memory (GEM)

Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017b) has a constrained optimization-based update strategy. While updating the current task loss Eq. (4.2), GEM ensures that losses on the episodic memory (of  $k < t$  tasks) Eq. (4.3) does not increase in comparison to the previous task model ( $\theta_{t-1}$ ). Formally, the constrained objective is defined as follows:

$$\min_{\theta} L_{TASK}(\theta; \mathcal{D}_t) \quad \text{s.t.} \quad L_{REP}(\theta; \mathcal{M}_k) \leq L_{REP}(\theta_{t-1}; \mathcal{M}_k) \quad \forall k < t. \quad (4.6)$$

To inspect the episodic memory loss increase, GEM computes the angle between the loss gradient vectors of previous tasks  $g_k$  and the proposed gradient update on the current task  $g$ . When the angle between  $g$  and any of the  $g_k$ 's is greater than  $90^\circ$ ,  $g$  is projected to the closest in  $\ell_2$ -norm gradient  $\tilde{g}$  such that it avoids the increase in losses but allows their decrease. Formally, the modified objective is defined as follows:

$$\min_{\tilde{g}} \frac{1}{2} \|g - \tilde{g}\|_2^2 \quad \text{s.t.} \quad \langle \tilde{g}, g_k \rangle \geq 0 \quad \forall k < t. \quad (4.7)$$

GEM solves the above optimization problem Eq. (4.7) via quadratic programming in the dual space with  $t - 1$  variables ( $v \in \mathbb{R}^{(t-1) \times 1}$ ):

$$\min_v \frac{1}{2} v^\top G G^\top v + g^\top G^\top v \quad \text{s.t.} \quad v \geq 0, \quad (4.8)$$

where  $G = -(g_1, \dots, g_{t-1}) \in \mathbb{R}^{(t-1) \times P}$ ,  $g \in \mathbb{R}^{P \times 1}$ ,  $P$  is the number of model parameters. Notice that  $G$  is computed at each gradient step of training. Let  $v^*$  denote the solution of Eq. (4.8), then the projected gradient used for updating the model is computed as  $\tilde{g} = G^\top v^* + g$ . Under the unified framework Eq. (4.4), GEM algorithm reduces to setting the relative importance weights  $\alpha_1(\theta_t^k)$  and  $\alpha_2(\theta_t^k)$  as follows:

$$\alpha_1(\theta_t^k) = 1, \quad \alpha_2(\theta_t^k) = v^* \quad (4.9)$$

### 4.1.3 Averaged Gradient Episodic Memory (A-GEM)

GEM constraints the current task gradient such that the episodic loss on each of the previous tasks  $k < t$  Eq. (4.3) does not increase. To

enforce these constraints, it requires computing the gradient using the whole replay buffer  $\mathcal{M}$ , as well as solving a quadratic programming problem Eq. (4.7). However, the expensive nature of these computations limits the scalability of the GEM to a large number of tasks.

Averaged GEM (A-GEM) provides a more efficient version of GEM by relaxing the constraints as it only requires that the *average* episodic memory loss over the previous tasks does not increase, which reduce the constraints from  $t - 1$  constraints to a single constraint:

$$\min_{\theta} L_{TASK}(\theta; \mathcal{D}_t) \quad \text{s.t.} \quad L_{REP}(\theta; \mathcal{M}) \leq L_{REP}(\theta_{t-1}; \mathcal{M}) \quad \text{where } \mathcal{M} = \cup_{k < t} \mathcal{M}_k. \quad (4.10)$$

The optimization problem corresponding to Eq. (4.10) is defined as:

$$\min_{\tilde{g}} \frac{1}{2} \|g - \tilde{g}\|_2^2 \quad \text{s.t.} \quad \tilde{g}^\top g_{REP} \geq 0, \quad (4.11)$$

where  $g_{REP}$  is a gradient computed using batch of replay examples, sampled randomly over all previously seen tasks from the episodic memory. Eq. (4.11) can be solved using just an inner product between the gradients of  $L_{TASK}$  ( $g$ ) and  $L_{REP}$  ( $g_{REP}$ ) instead of a quadratic program. When the current task gradient  $g$  violates the constraint, the project gradient  $\tilde{g}$  is computed as:

$$\tilde{g} = g - \frac{g^\top g_{REP}}{g_{REP}^\top g_{REP}} g_{REP}. \quad (4.12)$$

For the composite objective Eq. (4.4), A-GEM algorithm reduces to setting the importance weights  $\alpha_1(\theta_k^t)$  and  $\alpha_2(\theta_k^t)$  as follows:

$$\alpha_1(\theta_k^t) = 1, \quad \alpha_2(\theta_k^t) = \mathbf{I}_{\langle g, g_{REP} \rangle \leq 0} \left( - \frac{g^\top g_{REP}}{g_{REP}^\top g_{REP}} \right), \quad (4.13)$$

where  $\mathbf{I}_A$  is the indicator function which evaluates to 1 if A holds and otherwise to 0.

#### 4.1.4 Mixed Stochastic Gradient (MEGA)

Under the unified framework, one can see that GEM and A-GEM put the same weight on the current task loss regardless of how the loss

changes over time ( $\alpha_1(\theta_k^t) = 1$  in Eq. (4.9), Eq. (4.13)). Guo *et al.* (2020) argues that such a strategy does not capture a good balance between current task loss Eq. (4.2) and replay loss Eq. (4.3). For example, if the current task loss is small ( $L_{TASK} < \epsilon$ ), then the model performs well on the current task, and the model should focus on previous tasks in the episodic memory. On the other hand, if the current task loss is larger ( $L_{TASK} > \epsilon$ ), then the algorithm should weigh the current task loss relatively higher compared to the replay loss. Based upon this intuition, Guo *et al.* (2020) proposes Mixed Stochastic Gradient (MEGA-I), which adaptively balances two losses by leveraging the loss information available during training. For the unified update Eq. (4.5), MEGA-I sets

$$\begin{aligned} \alpha_1(\theta_k^t) = 1, \quad \alpha_2(\theta_k^t) &= \frac{L_{REP}(\theta_k^t)}{L_{TASK}(\theta_k^t)} && \text{if } L_{TASK}(\theta_k^t) > \epsilon \\ \alpha_1(\theta_k^t) = 0, \quad \alpha_2(\theta_k^t) &= 1 && \text{if } L_{TASK}(\theta_k^t) \leq \epsilon \end{aligned} \quad (4.14)$$

where  $\epsilon$  is a pre-defined threshold parameter.

#### 4.1.5 Meta-Experience Replay (MER)

Riemer *et al.* (2019) proposed learning to learn technique using gradient alignment (similar to GEM (Lopez-Paz and Ranzato, 2017b)) to reduce backward interference with a possibility of future transfer. MER also maintains an experience replay style memory with reservoir sampling. They optimize for the following objective to maximize transfer and minimize interference in lifelong learning:

$$\min_{\theta} \mathbb{E}_{B_t \sim \mathcal{M}} \left[ \sum_{t=1}^T \left[ L(\theta; B_t) - \sum_{p,q \leq t} \alpha \frac{\partial L(\theta; B_p)}{\partial \theta} \frac{\partial L(\theta; B_q)}{\partial \theta} \right] \right] \quad (4.15)$$

where  $B_t$  is the batch of data points belonging to a task  $t$ ,  $L$  is the loss function,  $\mathcal{M}$  is the memory and  $T$  is the total number of tasks. In other words, the authors integrate the Reptile algorithm (Nichol *et al.*, 2018) (that was defined in meta-learning context) with an experience replay module to help in lifelong learning by discovering notions of tasks without supervision.

Few recent papers that use experience replay in the lifelong learning domain include Look-ahead MAML (La-MAML) (Gupta *et al.*, 2020), another meta-learning method that modulates per-parameter learning rates to pace the learning of a model across tasks and time. Batch-level Experience Replay (Mai *et al.*, 2020) modifies ER mainly by performing a review step before the final testing to remind the model of the knowledge it has learned during the whole training. Apart from that, Buzzega *et al.* (2021) modify ER by applying several tricks like performing augmentation, adding a bias correction layer in the model, decaying the learning rate exponentially, and sampling examples greedily using the training loss value.

## 4.2 Test-time use of Episodic Memory

Section 4.1 focuses on approaches that use episodic memory during training. However, lifelong learning is a continuous process, and there might not be a clear delimitation between training and evaluation. So one can assume access to the episodic memory even during evaluation. Following this assumption, several works propose to use episodic memory during evaluation (Rebuffi *et al.*, 2017; Sprechmann *et al.*, 2018; Masson d’Autume *et al.*, 2019; Wang *et al.*, 2020b) and we will review canonical methods in detail here.

### 4.2.1 Incremental Classifier and Representation Learning (iCaRL)

iCaRL (Rebuffi *et al.*, 2017) is among the first methods to use episodic memory during test-time. iCaRL introduces three components to alleviate the catastrophic forgetting in the class incremental learning setup: (i) representation learning using knowledge distillation (Hinton *et al.*, 2015) and experience replay (section 4.1.1), (ii) herding based example selection for MEMWRITE, and (iii) test-time classification using nearest mean-of-features from episodic memory.

**Training.** The idea of using knowledge distillation is similar to the Learning without Forgetting (LwF) approach we discussed in Section 3.4.1. Basically, after introducing a new set of classes, the distillation

loss is evaluated on the older classes to ensure that the outputs of the current model are close to the output of the previous model, and the classification loss is evaluated only on the new classes. Next, the iCaRL uses a herding-based example selection strategy to write examples to the episodic memory. Instead of choosing the examples randomly, herding chooses them such that their mean approximates the class mean in the feature space. Formal discussion about the herding approach is deferred to Section 4.3.

**Inference.** Let us denote  $f_{emb}$  to be a feature extractor and  $g_w$  to be an output layer (e.g., linear layer followed by a Softmax). Under the parametric model, the prediction rule is given as  $y^* = \arg \max_{y \in 1 \dots C} g_w(f_{emb}(x)) = \arg \max_{y \in 1 \dots C} w_y^\top f_{emb}(x)$ . This prediction rule can be viewed as a linear classifier (with weights  $w_1, \dots, w_C$ ) on top of non-linear features (from  $f_{emb}$ ). Now in the context of class-incremental learning setup, our weights  $w$  are decoupled from the underlying feature extractor  $f_{emb}$ . As a result of this, whenever  $f_{emb}$  changes, then corresponding predictions can go unchecked, leading to a severe performance drop (interpreted as forgetting). To overcome this issue, iCaRL suggests using episodic memory for classification. Specifically, they use the nearest mean-of-exemplars classification strategy to predict a label  $y^*$  for an example  $x$ . First, a prototype vector for each class is computed, and then it is used to predict the class label with the most similar prototype with  $f_{emb}(x)$ . The prototype for class  $y$  is defined to be the average feature vector of all examples with class label  $y$  in the episodic memory. Formally, the prediction rule is as follows:

$$y^* = \arg \min_{y=1, \dots, C} \left\| f_{emb}(x) - \frac{1}{|M(y)|} \sum_{x' \in M(y)} f_{emb}(x') \right\|, \quad (4.16)$$

where  $M^{(y)}$  denotes all examples with class label  $y$  in the episodic memory. As the  $f_{emb}$  changes, the class prototypes update accordingly, therefore, the prediction rule Eq. (4.16) does not suffer from decoupled weights issue.

### 4.2.2 Memory-based Parameter Adaptation (MbPA)

Inspired from CLS theory, Sprechmann *et al.* (2018) introduce a method that consists of two components: a parametric component (neural network) that learns slowly and a non-parametric component (episodic memory with instances from previous tasks) that rapidly adapts to the parametric component. Particularly, episodic memory is used for instance-based (local) adaptation of the parametric network at inference time. Hence, Sprechmann *et al.* (2018) term their approach as Memory-based Parameter Adaptation (MbPA). iCaRL proposes a nearest-mean-of-exemplars classifier to overcome issues relating to decoupled weights. MbPA can be viewed as an alternate way of updating the classifier at the test time to address the same issue.

The parametric component consists of an embedding network  $f_{emb}$  and a task network  $g_w$ . The embedding network is used to encode the instances and the task network is used to predict the output class distribution,  $p(y|x, emb, w) = g_w(f_{emb}(x))$ . Unlike the previous memory-based approaches, Sprechmann *et al.* (2018) stores instances in the form of key and value pairs, i.e.,  $M_t = \{(h_i, v_i)\}$ , where  $h_i = f_{emb}(x_i)$  and  $v_i = y_i$ . During training, the usual maximum likelihood estimation is used to estimate the parameters  $\{w, emb\}$ . Apart from populating the episodic memory with the observed examples, it is not used during training.

**Inference.** Similar to the iCaRL, MbPA uses episodic memory for classification. Particularly, the encoding of the current input  $f_{emb}(x)$  is used to retrieve  $k$  nearest neighbors from the episodic memory  $C = (h_i, v_i, w_i)_{i=1}^k$ . The weight  $w_i$  measures the closeness of the example to the  $f_{emb}(x)$  and is defined using the kernel function:

$$\text{kern}(h_i, f_{emb}(x)) = \frac{1}{\epsilon + \|h_i - f_{emb}(x)\|_2^2}. \quad (4.17)$$

The local adaptation component corresponds to adapting the output parameters  $w$  to minimize the weighted average negative likelihood over

the retrieved  $k$  neighbors. Formally, the update rule is defined as:

$$w^x = \arg \min_w - \sum_{i=1}^k w_i \log p(y_i | x_i; w). \quad (4.18)$$

Notice that the episodic memory contains keys from the embedding network at different points during the training. Masson d’Autume *et al.* (2019) argues that this results in the embedding network from drifting over time, and the key of the test examples is closer to that of the recently seen examples. To circumvent this issue, they suggest freezing the embedding network. Given the recent surge of generic pre-trained models, Masson d’Autume *et al.* (2019) initializes their embedding network with pre-trained transformer-based BERT model (Devlin *et al.*, 2018) for lifelong language learning.

As MbPA based approaches locally adapt the model at test-time, Wang *et al.* (2020b) argue that this results in train and test-time discrepancy as the model is never locally updated during train time. This discrepancy results in negative transfer when locally updated models are evaluated on test examples from the last task. To address this problem, Wang *et al.* (2020b) proposes an efficient meta-lifelong learning framework, Meta-MbPA, by recasting the local adaptation problem as learning to “quickly” remember using the episodic memory.

### 4.3 Memory Read & Write Sampling Strategies

In this section, we discuss several strategies for selecting which examples to store in the episodic memory. Some of these strategies are inspired by recent neuroscience research, while others are based on statistical insights. Gupta *et al.* (2010) suggest that humans replay infrequent events more often than frequent ones. Particularly, infrequent events deemed to be surprising (Cheng and Frank, 2008; McNamara *et al.*, 2014) or rewarding (Atherton *et al.*, 2015; Ólafsdóttir *et al.*, 2015). On the other hand, statistical strategies promote matching the data distribution of all tasks with that of the episodic memory (Bickel *et al.*, 2008; Rebuffi *et al.*, 2017). Some suggest (Aljundi *et al.*, 2019d; Wang *et al.*, 2020b) to maximize the coverage of the episodic memory by selecting diverse examples. Alternatively, Wang *et al.* (2020b) provide active learning



inspired view of sample selection strategies: a diversity-based method that picks the most representative examples and an uncertainty-based method that picks the most unsure examples (surprise (Ramalho and Garnelo, 2019), forgettable (Toneva *et al.*, 2019)). We will discuss a subset of these strategies in this section.

**Herding.** Rebuffi *et al.* (2017) propose to select  $m$  examples per class by iteratively selecting the examples that best approximate the average feature vector over all  $n$  training examples. This iterative example selection is called herding (Welling, 2009) and works in an offline manner, i.e., after training the model for the given class  $c$ . Due to the offline iterative selection strategy, resulting examples constitute a representative set of samples from a distribution. Thus, this strategy strives to match the distribution of examples with that of the episodic memory at the individual class level. Let  $f_{emb} : \mathcal{X} \rightarrow R^d$  be a feature extractor,  $D_c = \{x_1, \dots, x_n\}$  be the  $n$  examples corresponding to  $c^{th}$  class.

$$\mu_c = \frac{1}{|D_c|} \sum_{x_i \in D_c} f_{emb}(x_i), \quad (4.19)$$

$$M_j^c = \arg \min_{x_i \in D_c} \left\| \mu_c - \frac{1}{k} [f_{emb}(x_i) + \sum_{j=1}^{k-1} f_{emb}(x_j)] \right\|. \quad (4.20)$$

**Surprise.** Cheng and Frank (2008) and McNamara *et al.* (2014) discuss that replay in rodents is connected to unexpected events and based upon this inspiration, Isele and Cosgun (2018) propose a surprise criterion for sampling transitions in incremental reinforcement learning. On the other hand, Ramalho and Garnelo (2019) propose an algorithm to approximate the task distribution based upon the surprising examples encountered during training. Formally, surprise for an example is computed using the model’s prediction as  $S = -\log(y_t)$ . Intuitively, the higher the probability that the model assigns to the true label  $y_t$ , the less surprising that example is. Further, Wang *et al.* (2020b) investigate this approach in the context of lifelong language learning by viewing it as one of the uncertainty-based sample selection strategies.

**Reward.** Similar to surprise, some other neuroscience studies (Atherton *et al.*, 2015; Ólafsdóttir *et al.*, 2015) suggest that rewarding events are often associated with the replay. Therefore, in the context of incremental reinforcement learning, Isele and Cosgun (2018) propose a reward-based sample selection strategy. Concretely, the absolute value of the future discounted return is used to select the rewarding experiences,  $\mathcal{R}(e_i) = |R_i(e_i)|$ .

**Diversity / Coverage maximization.** Isele and Cosgun (2018) and Wang *et al.* (2020b) argue that when the memory buffer is limited in size, it is helpful to sample diverse examples to maximize coverage of the underlying data distribution. Isele and Cosgun (2018) propose to sample by ranking experiences based upon the number of neighbors in the episodic memory. The experience with the most number of neighbors is selected for replacements. Similarly, Wang *et al.* (2020b) leverage a pre-trained feature extractor for estimating the diversity of the sampled examples. Intuitively, for a given example, if there are nearest neighbors in the episodic buffer, that particular example is less diverse and sampled rarely (low probability). Concretely, given a feature extractor  $f_{emb}$ , episodic memory module  $\mathcal{M}$ , the probability for selecting example  $x'$  is defined as follows:

$$\log(p(x')) \propto \min_{x \in \mathcal{M}} \|g_{emb}(x') - f_{emb}(x)\|_2^2. \quad (4.21)$$

**Reservoir sampling.** Isele and Cosgun (2018) argue that the best strategy for sampling is the one that matches the distribution of the episodic buffer with that of the global train/test distribution over all tasks. In lifelong learning, we see online streams of data, and the global distribution is not known in advance. Therefore, most of the recent works resort to reservoir sampling (Vitter, 1985). Given an input stream with unknown length,  $m$  to be the maximum capacity of the buffer, and  $n$  to be the number of examples observed so far, reservoir sampling picks examples with the probability  $m/n$ .

Wang *et al.* (2020b) compares a representative set of the above-discussed strategies and finds that diversity-based sample selection strategies outperform uncertainty-based selection strategies. Notice that

reservoir sampling can be viewed as a diversity-based method since it picks examples representing the true data distribution.

**Gradient based sample selection.** All of the above sample selection methods make an implicit or explicit assumption about the availability of task boundary. However, we might not have access to the information in some scenarios when a particular task changes. Motivated by this scenario, Aljundi *et al.* (2019d) develop a gradient-based sample selection strategy to populate the replay buffer without any knowledge about the underlying task identity. Specifically, sample selection is formulated as a constraint reduction problem based on a constrained optimization view of the continual learning (see Section 4.1.2 for original formulation, Eq. (4.6) and Eq. (4.7)). From the original constraints in the gradient space ( Eq. (4.7)), Aljundi *et al.* (2019d) propose selecting examples so that the feasible region formed by the constraints corresponding to the selected subset of examples is close to that of the original region. Given previous  $t$  tasks ( $[0, \dots, t-1]$ ), the original feasible region ( $C$ ) and the reduced feasible region ( $\tilde{C}$ ) corresponding to the memory  $\mathcal{M}$  are defined as follows:

$$C = \bigcap_{i \in [0, \dots, t-1]} \{g | \langle g, g_i \rangle \geq 0\} \quad (4.22)$$

$$\tilde{C} = \bigcap_{g_i \in \mathcal{M}} \{g | \langle g, g_i \rangle \geq 0\}. \quad (4.23)$$

As  $\mathcal{M}$  is a subset of the previous tasks examples, the reduced feasible region  $\tilde{C}$  is infact larger than the original region  $C$  (the number of examples corresponds to the number of constraints defining the feasible region). Therefore, finding the smallest  $\tilde{C}$  suffices the criterion that  $\tilde{C}$  is close to  $C$ . To define the notion of closeness, the size of the feasible region (convex cone) is defined in terms of the solid angle between the cone and the unit sphere. Moreover, the number of constraints (gradients) is smaller than the dimension of the gradient. Therefore, the feasible region and the solid angle can be defined in  $M$ -dimensional space  $span(\mathcal{M})$ . Thus, the sample selection objective is defined as follows:

$$\min_{\mathcal{M}} \lambda_{M-1} \left( S_{M-1}^{\text{span}(\mathcal{M})} \cap \bigcap_{g_i \in \mathcal{M}} \{g | \langle g, g_i \rangle \geq 0\} \right), \quad (4.24)$$

where  $M = |\mathcal{M}|$ ,  $S_{M-1}^{\text{span}(\mathcal{M})}$  denotes a unit sphere in  $M - 1$  dimensional space, and  $\lambda_{M-1}$  is Lebesgue measure. As the above optimization problem is hard to minimize, Aljundi *et al.* (2019d) propose a surrogate to Eq. (4.24). Based on the observation that one can reduce the feasible region by increasing the angle between each pair of gradients, the surrogate objective for sample selection is defined as follows:

$$\min_{\mathcal{M}} \sum_{i,j \in \mathcal{M}} \frac{\langle g_i, g_j \rangle}{\|g_i\| \cdot \|g_j\|} \text{ s.t. } \mathcal{M} \subset [\mathcal{D}_1, \dots, \mathcal{D}_{t-1}]. \quad (4.25)$$

**Gradient-based diversity maximization.** Interestingly, Aljundi *et al.* (2019d) show that minimizing the above objective corresponds to maximizing the variance of the gradient direction,  $\text{Var}[\hat{g}]$ , where  $\hat{g}$  is a unit vector.

$$\text{Var}_{\mathcal{M}}[\hat{g}] = \frac{1}{M} \sum_{k \in \mathcal{M}} \|\hat{g}_k\|_2^2 - \left\| \frac{1}{M} \sum_{k \in \mathcal{M}} \hat{g}_k \right\|_2^2 = 1 - \frac{1}{M^2} \sum_{i,j \in \mathcal{M}} \frac{\langle g_i, g_j \rangle}{\|g_i\| \cdot \|g_j\|}. \quad (4.26)$$

Previously, we looked at a diversity-based sample selection strategy where diversity is defined in terms of the hidden representations as features. Gradient-based sample selection surrogates can be interpreted as selecting diverse examples based on gradients as features.

**Forgettable.** Instead of analyzing the model performance under distributional shift, Toneva *et al.* (2019) investigate the learning dynamics of neural networks on a single task. Specifically, Toneva *et al.* (2019) define the occurrence of a forgetting event when the model transitions from correct to incorrect classification for individual training examples. They report that different examples are forgotten at different frequencies, and removing a significant fraction of least forgettable examples from

training data still results in competitive performance. On the other hand, forgettable examples have uncommon features and are difficult to classify. Inspired by these findings, Wang *et al.* (2020b) studies the effectiveness of forgettable examples for replay by considering it as one of the uncertainty-based sample selection strategies.

**Hindsight anchor learning.** The forgettable examples are sampled from the actual observations in the above method. On the other hand, Chaudhry *et al.* (2021) propose to explicitly construct pseudo examples/anchors such that the anchors undergo maximum forgetting after training on future tasks. Formally, given current task  $t$ , the desirable anchor  $a_t$  with label  $y_t$  can be obtained by maximizing the following loss:

$$(a_t, y_t) \leftarrow \arg \max_{(x, y) \sim P_t} \ell(f_{\theta_T}(x), y_t) - \ell(f_{\theta_t}(x), y_t), \quad (4.27)$$

where  $\theta_T$  is the model after training on the future task  $T (> t)$ . However, for the above optimization problem, one requires access to the entire distribution  $P_t$  and future tasks. To avoid storing the entire dataset for estimating  $P_t$ , Chaudhry *et al.* (2021) maintain the running average of the mean feature embedding  $f_{emb}^t$  as follows:

$$f_{emb}^t \leftarrow \beta f_{emb}^t + (1 - \beta) \frac{1}{|b_t|} \sum_{x \in b_t} f_{emb}(x). \quad (4.28)$$

As we do not have access to future tasks, Chaudhry *et al.* (2021) suggest approximating the future by simulating the past, i.e., evaluate forgetting of the current task after fine-tuning on the past tasks. Hence, the modified objective to learn maximal forgettable anchor is defined as:

$$(a_t, y_t) \leftarrow \arg \max_{a_t \in \mathbb{R}^D} \ell(f_{\theta_M}(x), y_t) - \ell(f_{\theta_t}(x), y_t) - \gamma (f_{emb}(a_t) - f_{emb}^t)^2. \quad (4.29)$$

As the above method evaluates forgetting in hindsight, the method is called hindsight anchor learning.

## 4.4 Generative Replay

As mentioned earlier, the CLS theory (McClelland *et al.*, 1995; O'Reilly and Norman, 2002) proposes that human memory consists of dual complementary systems: one for gradual accumulation of the structured knowledge (neocortex) and another one for rapid encoding of the current inputs (hippocampus). Moreover, the hippocampal system reactivates the memory trace during sleep (Stickgold and Walker, 2007) for the long-term memory consolidation in the neocortex with the help of multiple replays of the encoded experiences. In line with this mechanism, the memory-based approaches retain examples from past tasks for replaying them to alleviate forgetting. Further, there are pieces of evidence (Stickgold and Walker, 2007; Ramirez *et al.*, 2013) that the hippocampal system also generates false memory experiences while replaying, thus, performing more than a naive replay. Based upon these studies, Shin *et al.* (2017) argue that generative models are better conceptualizations of the hippocampal system than the replay buffer. Further, one of the issues with simply replaying of examples from past tasks is that it requires a large memory, which is often unrealistic in real-world applications where access to the past tasks' data is limited (privacy concerns). By considering the generative models of the data, one can generate pseudo-data for experience replay, thus, relaxing the need to retain the actual examples. In this primer, we discuss two canonical works along this line, (1) Shin *et al.* (2017) propose a deep generative replay framework with a generative adversarial network (GAN) to mimic the past data and studies the problem on image classification tasks, and (2) Sun *et al.* (2020) introduce a language model that simultaneously learns to solve the task and generate pseudo-samples of previous NLP tasks.

**GAN framework.** Generative models learn to generate realistic samples by maximizing the likelihood of generated samples being in a given data distribution. GAN is one such kind of generative model that defines a zero-sum game between a generator network (G) and a discriminator network (D). The discriminator learns to distinguish between the real and the generated samples, while the generator learns to mimic the

given data distribution so that it can fool the discriminator. Formally, given the real data distribution  $p_{data}$ , the overall objective for both the networks is defined as follows:

$$\min_G \max_D V(D, G) = \mathbf{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbf{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] . \quad (4.30)$$

#### 4.4.1 Continual Learning with Deep Generative Replay

Based upon the above GAN framework, Shin *et al.* (2017) propose a scholar model  $H$  consisting of a generator  $G$  and a solver  $S$ . Given a new task  $t$ , a scholar model  $H_t$  is trained in two stages using the task  $t$ 's data and a previous scholar model  $H_{t-1}$ . During the first stage, a generator  $G_t$  learns to reconstruct the current data ( $x \sim p_t$ ) and the past data (pseudo-samples from  $G_{t-1}$ ). In the next stage, a solver  $S_t$  learns to solve the given task  $t$  while remembering the previous tasks (pseudo-labels  $\hat{y} = S_{t-1}(x)$ ). The overall objective for the scholar model  $H_t$  is as follows:

$$L(H_t) = r \mathbf{E}_{(x,y) \sim p_t} [L(S_t(x), y)] + (1 - r) \mathbf{E}_{x \sim G_{t-1}} [L(S_t(x), S_{t-1}(x))] , \quad (4.31)$$

where  $r$  is the mixing coefficient for the two objectives.

Another approach (Ven *et al.*, 2020), motivated by anatomy, modifies standard generative replay by merging the generator into the main model. This allows replaying the hidden representations that are generated by the model's context-modulated feedback connections.

#### 4.4.2 LAnguage MOdeling for Lifelong Language Learning (LAMOL)

McCann *et al.* (2018) show that multiple NLP tasks can be cast to a unified question-answering task, thereby enabling the use of a single language model (LM) to solve multiple tasks, i.e., given the context and question, the language model generates an answer. Based upon this observation, Sun *et al.* (2020) explore language modeling for lifelong language learning (LAMOL). Fundamentally, LM is inherently a text generator and can learn to generate samples from previous tasks. Taking

inspiration from the deep generative replay, Sun *et al.* (2020) propose to continuously train a pre-trained language model that simultaneously answers the questions and generates pseudo-samples of the previous tasks.

Although the generative replay-based approaches learn a single model without retaining old task examples, their performance is strictly upper bounded by the replay-based approaches that retain at least a few examples in the buffer (Sun *et al.*, 2020). There are open questions around the scalability of the generator with the number of tasks and potential conflicts between the generator and downstream tasks due to fixed shared model capacity.

## 4.5 Summary

In this chapter, we presented the unified view of memory-based methods and algorithms in lifelong learning. These methods maintain an episodic memory, containing a few examples from past tasks, and revisit it while learning a new task. We saw in regularization-based methods that different ways to penalize drastic changes in the model parameters were employed and incorporated into the overall objective. Along similar lines, memory-based methods are realizations of three primary strategies combined: 1) how to sample examples from memory, 2) how to update the model with current task loss along with the replay memory loss, and 3) how to select examples to write to the memory.

Most memory-based methods involve defining specific model update strategies using episodic memory for training the model on each new task. In addition, we also presented approaches that use episodic memory during test time for evaluation to prevent catastrophic forgetting.

Next, we presented numerous read/write sampling strategies employed by lifelong learning researchers. Several statistical sample selection strategies are inspired by areas like CLS theory, reinforcement learning, neuroscience and can be classified as diversity-based and uncertainty-based.

While memory-based methods retain examples from past tasks for replaying, generative replay methods avoid storing the examples. Instead, these methods take inspiration from neuroscience and the anatomy of



the human brain and generate pseudo-data for experience replay. These methods learn a single model to generate replay data that mimic the actual examples.

So far, we have presented lifelong learning methods that assumed a fixed capacity of the ML model. In the next chapter, we will discuss methods based on isolating task-specific parts of the model and even modifying its architecture to avoid interference for training on diverse tasks.

# 5

---

## Architecture-based Approaches

---

In this chapter, we will study the different architecture families (and their instantiations) that have been proposed for training lifelong learning systems. Here, *architecture* refers to the overall structure of a training system, and *architecture family* refers to a family of related architectures. For example, ResNet18, ResNet32, ResNet50, ResNet152 (He *et al.*, 2016), and WideResNet32 (Xie *et al.*, 2017) are architectures that belong to the family of residual networks. We organize this chapter in terms of the different architectural families.

Some of these architectures are general-purpose and can be used with different settings (for example, class-incremental or task-incremental), modalities (for example, computer vision or natural language), tasks (for example, classification or regression), and datasets. Other architectures are more specialized and useful in specific setups and scenarios. Essentially, the less general approaches make additional assumptions about the setup. If the assumptions hold for the given setup, these approaches are expected to perform better than the general architectures. The choice between general-purpose and specialized architectures is often driven by how much information we have about the setup; the more information we have, the more specialized architecture can be used.

We will start the discussion with the general-purpose architectures and introduce the specialized architectures along the way.

Architecture-based approaches can be seen as a mechanism to provide useful inductive biases to the learning system. For example, when the system is trained on a sequence of closely related tasks, it may be helpful to infer the changes across the tasks as new tasks are encountered. For example, the first task could be to “pick up a ball” and the second task could be to “pick up a cube”. Knowing “what changes across tasks” enables the use of knowledge (from previous tasks) to train the system for the incoming task. The underlying idea is that since we have some additional information about the problem setup (for example, the tasks are closely related), we can design an architecture that can leverage the extra information. We note that while we are focusing on architectures in this chapter, in practice, these architectures are often used in conjunction with other approaches such as regularization-based methods like elastic weight consolidation (as discussed in Chapter 3) or memory-based methods like experience replay (as discussed in Chapter 4). Despite the complementary nature of architecture-based approaches, we study them in a separate chapter to understand the common principles and motivations behind these architectures while abstracting away some details like the different replay mechanisms available. The rest of the chapter is organized as follows: We start with Modular Networks, the motivation behind their use, general architecture design, some common manifestations, and limitations. Next, we discuss the parameter isolation systems, which includes both fixed-capacity approaches like masking and pruning and dynamics-capacity approaches (Yoon *et al.*, 2018). We conclude with a discussion on some recently proposed approaches for lifelong learning on graphs. We include these approaches in this chapter since these approaches rely on the inductive biases specific to learning problems in the context of graphs.

## 5.1 Modular Networks

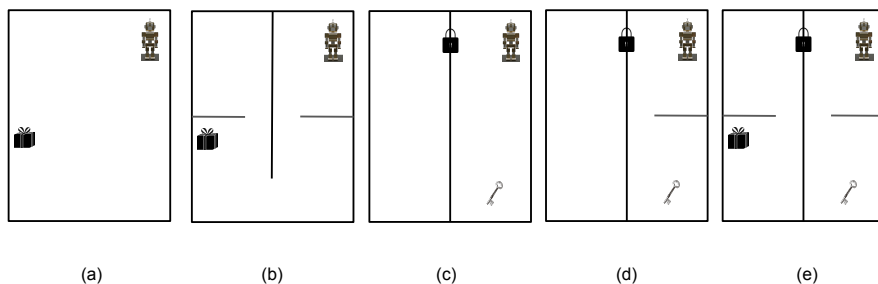
Much of the recent success of the machine learning models are limited to the *single-task* setups where the data distribution is well-defined and known beforehand. We know that as the system is trained on new tasks,

its performance on the previous tasks deteriorates (due to catastrophic forgetting). However, there are several challenges even before a new task is encountered. If the data distribution changes between training and evaluation, the learning system’s performance often diminishes drastically (Zhang *et al.*, 2018; Zhang *et al.*, 2020a; Cobbe *et al.*, 2020) suggesting that the system is not able to adapt to even small variations in the data distribution. This behavior is in stark contrast to how humans learn and operate. Not only are humans more *robust* learners, they are much more *sample efficient* and can quickly adapt to new tasks/data distributions.

Several hypotheses have been proposed to explain the discrepancy between the learning behavior of humans vs. machine learning systems. One of the more popular hypotheses is that the world is inherently compositional, i.e., the representation of the *whole* is composed of the representation of the *parts* and the humans exploit this compositionality to understand and operate in the world (Parascandolo *et al.*, 2018). This compositionality also implies that a *novel* task can be broken down into *parts* that we have already encountered in the previous tasks (in a different manifestation). For example, when reading a sentence, we break it down into phrases and words and derive meaning from it (even if we have never seen the same sentence before). Essentially, we exploit the compositionality of the world by learning modular, reusable, and general-purpose mechanisms (Goyal *et al.*, 2021) (or skills). These mechanisms can be shared across different tasks, thus making learning in humans more efficient than learning in neural networks. Since leveraging compositionality is useful for humans, it may be a useful inductive bias for developing lifelong learning systems that operate in the real world and make decisions over extended periods.

Another example for such a case is given in Fig. 5.1. To reach the goal (the gift box) in (e), the robot will have to learn to solve the first four tasks that involve walls (b), locked doors (c) and their composition (d). When the robot solves these relatively easier tasks, it can exploit the gained knowledge and reuse it to solve the final much harder task quickly.

*Modular Networks* are proposed as one of the promising direction to learn systems that can effectively leverage compositionality to learn more



**Figure 5.1:** Example of compositionality

efficiently (Happel and Murre, 1994; SHARKEY, 1996; SHARKEY, 1997; Auda and Kamel, 1998; Caelli *et al.*, 1999; Auda and Kamel, 1999; Andreas *et al.*, 2016b; Andreas *et al.*, 2016a; Johnson *et al.*, 2017; Santoro *et al.*, 2017; Yu *et al.*, 2018; Alet *et al.*, 2018a; Alet *et al.*, 2018b). Modular Networks incorporate *modularity* as the primary inductive bias. Modularity is the property of a system that it can be broken down into several relatively independent, replicable, and composable *modules* (or smaller networks) (Amer and Maul, 2019). Each module can be thought of as learning to solve a *subtask* (or part of a given task). In the context of compositional world hypothesis, a modular network can solve a given task by (i) breaking it into subtasks, (ii) using modules to solve the subtasks, and (iii) using the solutions of the subtasks to solve the given task. Thus modular networks can also be interpreted as factorizing knowledge into different modules. Some of the underlying subtasks may change when the task/data distribution changes, even though the high-level task may remain the same. In such a case, if the knowledge is appropriately factorized, only some modules will need to adapt/change (to account for the change in some subtasks), and the other modules can be used as-is. In practice, this would result in faster adaptation to the new distribution (Bengio *et al.*, 2019). For example, consider an system that is trained to “pick up a cup from a table”. The system could learn two modules - one for *reaching the table* and the other for *picking the cup*. If the task changes such that the height of the table is increased, then the system only needs to update the module corresponding to *picking the cup* as the subtask of reaching the table is

not changed.

The benefits of modularity can be easily extended to lifelong learning. As a system trains over a distribution of tasks, it could decompose the tasks into subtasks that are shared across the task distribution. When the system encounters a new task, it could break the task into a combination of novel and previously seen tasks. In that case, the system only needs to learn the novel subtasks, instead of learning the new task from scratch. Building upon the previous example of “pick up a cup from a table”, the next task could be “place a knife on the shelf”. Both these tasks require the ability to “move around”. If the system has learned a module for “moving around” (as part of the first task), it can use that module in the second task as well, thus enabling the positive forward transfer of knowledge. Moreover, since the module for “moving around” may also be improved while training on the second task, it can potentially lead to a positive backward transfer of knowledge where training on the second task improves the performance on the first task.

We note that similar to modular networks, several other areas like Out-of-Distribution (OOD) generalization (Wang *et al.*, 2021), zero-shot generalization (Purushwalkam *et al.*, 2019), few-shot generalization, etc., also focus on narrowing the gap in the performance of humans and machine learning systems.

### 5.1.1 Motivation

The use of modular networks for lifelong learning can be motivated from various other perspectives as well.

1. **Cognitive Science Perspective:** Spelke (1990), Pinker (1994), Pinker (2005), Spelke and Kinzler (2007), and Xu *et al.* (2009) hypothesized several theories to explain how humans learn coherent, abstract, and highly structured representations of the world from the fragmented but concrete instances of *experiences*. The “theory theory” (Carey, 1985; Gopnik, 1988; Wellman and Gelman, 1992) states that as humans interact in the world, they construct intuitive theories of the world. These theories have three key aspects: (i) They involve coherent, abstract, causal representations of the world. (ii) They have distinct cognitive functions.

For example, theories enable both prediction of the future as well as counterfactual inferences. (iii) They have distinctive dynamic features. For example, the theories can be updated as humans undergo novel experiences and discover new knowledge. These different theories aim to explain the interplay between abstract knowledge and concrete knowledge. In the modular networks, the network topology can be seen as a manifestation of the abstract knowledge, and the modules can be seen as encapsulating the concrete knowledge.

2. **Evolutionary Perspective:** Kashtan and Alon (2005) and Kashtan *et al.* (2007) hypothesized that environments with Modularly Varying Goals (MVGs), i.e., environment consisting of varying goals with common subgoals, leads to modular networks. Clune *et al.* (2013) hypothesized that modularity evolves as a byproduct from selection to reduce connection costs (like creating, sustaining connections) in a network. This modularity is then sustained by the Modularly Varying Goals (MVGs). A lifelong learning system is expected to learn (and retain the knowledge of) a series of tasks over its lifetime. Modularity can be a useful inductive bias for the neural network if these tasks share some common substructure.
3. **Empirical Perspective:** From a practical perspective, a modular neural network can be interpreted as a system of modules where each module is designed to solve one specific task, and the controller learns the mapping between the tasks and the modules. If multiple tasks share a common subtask, the modules corresponding to these tasks can share knowledge with each other. When new tasks are encountered, the network will eventually run out of capacity, a problem referred to as *capacity saturation* (Sodhani *et al.*, 2020). Modular neural networks provide a workaround for that by enabling the addition of new modules that can be added to the system, without disrupting the existing modules (and the knowledge encoded by them). Modularity also helps with catastrophic forgetting by localizing the forgetting effect, i.e., forgetting knowledge about a task should only affect the modules related to that task and not the other modules.

### 5.1.2 Architecture

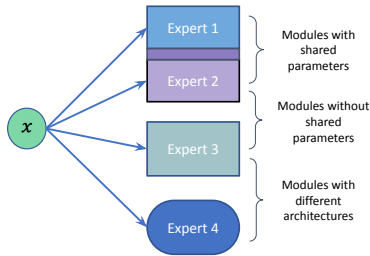
The high-level architecture of modular neural network can be described in terms of the following two components:

1. A system of  $n$  modules denoted as  $M = \{m_i \forall i \in \{1, \dots, n\}\}$ .
2. A *controller* mechanism that is used to decide the topology of connection between the modules.

We note that different works use different terminology for describing the architecture of neural modular networks. For example, some works denote the modules as *experts* (Rannen *et al.*, 2017; Aljundi *et al.*, 2017) or as *primitives* (Frans *et al.*, 2017; Goyal *et al.*, 2020) etc. Similarly, some works refer to the *controller* as the *gating mechanism* (Rannen *et al.*, 2017; Aljundi *et al.*, 2017) or as *router* (Rosenbaum *et al.*, 2017). While these different works have subtle differences in terms of how modules/controller are instantiated (or interact), the terminology we use here suffices to study them from the perspective of lifelong learning.

**System of Modules.** The first key component of modular networks is the system of  $n$  modules  $M$ . Each module  $m_i (\forall i \in \{1 \dots n\})$  is a neural network with parameters denoted as  $\theta_i$ . In terms of the network architecture, the modules may be identical (for example, a collection of ResNet50 models) or similar (for example, a collection of models from the ResNet family) or different (for example, some modules from the ResNet family and some modules from the VGG family). In the case of modules belonging to different architectures, the modules may also operate on different modalities. In terms of functional representation, the modules may be explicitly trained to encode different aspects of the input (maybe by providing direct supervision), or the modules may learn different representations on their own. Some modules may also share parameters with other modules (we denote the shared parameters of the  $i^{th}$  module as  $\theta_i^{shared}$ ). An example of a system of modules is shown in Fig. 5.2.





**Figure 5.2:** Different kind of modules in Modular Architectures

**Controller Mechanism.** The second key component of modular networks is the controller mechanism. A controller is any function  $C$  that defines a network topology using the system of modules. The input to the controller mechanism could include: input data points, metadata in the form of the task description, output from the system of modules, or a combination of these. Note that this description of the controller includes both implicit and explicit controllers (parameterized as well as non-parameterized). The output of the controller is a network topology  $z$ . This topology defines the *arrangement* of (or interaction between) modules, thus instantiating a function  $f_z$  as shown in Eq. (5.2). Note that all the modules are not required to be part of the network topology. Moreover, the controller may not have to learn the entire topology as some parts of the topology may be pre-determined. The controller may generate the entire network topology at once (Aljundi *et al.*, 2017) or step by step (as done in Goyal *et al.* (2021)).

$$z = C(x, \text{metadata}, M), \quad (5.1)$$

$$y = f_z(y|x, M). \quad (5.2)$$

**Training Mechanism.** Several training mechanisms have been proposed for training Neural Modular Networks. These mechanisms vary in terms of the following dimensions:

1. Should the modules and the controller be trained jointly, in an end-to-end manner (as done in for example Chang *et al.* (2019)) or

have separate losses for the two components. Should the controller be meta-trained?

2. Is the network layout  $z$  a discrete variable, like a hard-attention mask (or graph with 0-1 edges), or is the network layout represented as a continuous variable, like a soft-attention mask (or graph with soft-edges).
3. Is the controller mechanism explicit or implicit?
4. Is a per-module loss available?

**Relationship with standard Deep Learning architectures.** The standard, monolithic, deep learning architectures (like ResNets) can be seen as a special form of modular networks. The layers (of the monolithic network) can be seen as the modules, and the controller mechanism is the pre-determined and hardcoded topology where the output of one layer feeds into the next layer. In this sense, the process of learning modules is similar to the process of training layers in a neural network.

### 5.1.3 Lifelong Learning

Modular Networks represent a very general and flexible family of neural network architectures. They provide several benefits like the ability to use different model architectures as constituent modules and ability to add new modules throughout training. Modular neural networks are also biologically inspired (Bertolero *et al.*, 2015) and the brain has been shown to be modular at different spatial scales, from the micro level of synapses to the macro level of brain regions (Ihmels *et al.*, 2002; Newman, 2006; Chen *et al.*, 2008; Wagner *et al.*, 2007; Bullmore and Sporns, 2009; Bassett *et al.*, 2010; Meunier *et al.*, 2010). In the context of deep learning, Modular Network initially focused on visual questions answering (Andreas *et al.*, 2016b). Since then, they have been extended to several settings like multitask learning (Zhang *et al.*, 2020b; Sodhani *et al.*, 2021), compositional generalization (Goyal *et al.*, 2020; Goyal *et al.*, 2021; Chang *et al.*, 2019), etc. Different applications of modular networks, in context of lifelong learning, differs in terms of what the modules learn (or should learn).

**Modules can represent composable skills.** Several works (Frans *et al.*, 2017; Chang *et al.*, 2019; Vezhnevets *et al.*, 2017; Goyal *et al.*, 2020) use the modules to learn composable *skills*. The general idea is the following: When a model is trained on a given task, the model *learns* skills that it composes to solve the given task. The benefit of learning skills is that skills can be selectively transferred across tasks. In this case, the controller mechanism should help to ensure that modules learn a diverse set of skills and the model learns to solve a task by choosing and composing a subset of modules (skills).

Using modules for learning skills has an additional benefit: A new task may require learning new skills. In that case, new modules can be instantiated and trained on the given task (and used for the subsequent tasks). Some common challenges in this setup are: (i) learning a diverse set of skills, (ii) not forgetting the previously acquired skilled (in principle, this can be easily achieved by not finetuning a module on subsequent tasks, but this may force the model to learn very similar skills), (iii) controlling the growth of the number of modules (if the tasks share the skills, the number of modules should grow sub-linearly with the number of tasks).

We explain the general design and implementation of works in this category using Veniat *et al.* (2021) that proposed a neural modular network architecture for lifelong learning where each module  $m_i$  represents a composable, atomic skill. The model solves a task by transforming the given task to a new task: searching through an exponentially large search space (of the composition of modules) and inferring a composition that enables the model to solve the given task. While modules are shared across tasks (to enable knowledge transfer), modules corresponding to the older tasks are not updated for the newer tasks, thus avoiding catastrophic forgetting. New modules are added only when new skills are required. If tasks share skills, the module grows sub-linearly with the number of tasks.

At the start of training, the model is initialized as a collection of  $l$  modules, arranged in  $l$  layers with one module per layer. The modules can be different across the layers. During training, new modules could be added across the layers, with the constraint that all the modules in a layer are of the same architecture (but can learn different weights).

The setup can be explained with an example where the model has been trained on  $t - 1$  tasks. Now, when the  $t^{\text{th}}$  task arrives, a new randomly initialized module is added to each layer, and a search space is defined over all the possible ways to combine the modules (both old and new).

The resulting search space is exponentially large, and several constraints are imposed to keep the search tractable:

1. Modules within the same layer do not connect with each other, so only one module can be selected from each layer.
2. A newly added module, say at the  $i^{\text{th}}$  layer, can only connect to another newly added module at the  $i + 1^{\text{th}}$ .

The second restriction is motivated by the need to reduce the size of the otherwise exponentially large search space. In practice, this restriction can be justified as follows: as new tasks are added, changes are expected in the output distribution and not the input distribution. If the tasks are related, initial layers can be shared across the tasks.

The training objective is to minimize the loss corresponding to the  $t^{\text{th}}$  task, as a function of the parameters (of all the modules) and connection between the modules. This connection is essentially a path in the grid of modules. The connection of the  $t^{\text{th}}$  task is denoted as  $\pi_t$  and the parameters of the modules, that are part of this path, are denoted as  $\theta(\pi_t)$ . The model optimizes the loss function:

$$\Gamma^*, \theta^* = \arg \min_{\theta, \Gamma} \mathbb{E}_{j \sim \Gamma, (x, y) \sim \mathcal{D}^t} \mathcal{L}(f(x, t | \mathcal{S}, \theta(\pi_j)), y). \quad (5.3)$$

Here  $\mathcal{S}$  is the set of tasks model has seen so far,  $x, y$  denote the input and the target label for the  $t^{\text{th}}$  task,  $f(x, t | \mathcal{S}, \theta(\pi_j))$  is an instance of modular network using the path  $\pi_j$ ,  $\mathcal{L}$  is the loss function and  $\Gamma$  is a distribution over the set of possible paths. Other works have considered different approaches for selecting a path. For example Rajasegaran *et al.* (2019a) randomly select a path connecting the modules and keeps using that path until it saturates (i.e., the parameters have been fully used for learning on the previous tasks), while Fernando *et al.* (2017) use genetic algorithms to select a path connecting the modules.

Note that only the newly added modules can be trained, and the parameters of the previously added modules remain unchanged. At the

end of training on the  $t^{\text{th}}$  task, any new module, that does not appear on the optimal path, is not retained for the subsequent tasks, thus keeping the model’s growth sublinear in the number of tasks. The resulting path is saved for the given task and can be retrieved during testing.

There are two approaches for optimizing the loss in Eq. (5.3). In the first approach, called the stochastic variant, the model alternates between optimizing the path and optimizing the parameters of a given path. In terms of general modular network architecture, this corresponds to alternatively optimizing the controller mechanism and the modules. Specifically, the distribution  $\Gamma$  is modeled by a product of multinomial distributions, one for each layer of the model. The modules are selected *one-layer-at-a-time*. An entropy regularizer is used to encourage the model to explore different paths. The second approach, called the deterministic variant, is more straightforward - an exhaustive search is performed over the set of paths.

Veniat *et al.* (2021) uses a *data-driven prior* which works as follows: First,  $k$  tasks, which are most similar to the given task, are selected from the set of previous tasks. The search space for the current task is limited to the perturbations of the paths corresponding to these previous tasks. The most similar previous tasks are chosen by computing the predictions on the current task data, using paths from all the previous tasks (specifically, using the feature from the penultimate layer). The paths that yield the best nearest neighbor classification accuracy are selected. Several other mechanisms are also used to restrict the search space over paths further. For example Andreas *et al.* (2016b) selects only one module at a time (instead of selecting an arbitrary number of modules at any time), while Goyal *et al.* (2021) attends to all the modules.

#### 5.1.4 Limitations

There are two key limitations to existing approaches for training modular neural networks which limit their usability in lifelong learning applications:

1. Selecting the right level of *modularity*: One big open problem is how to enable the models to learn modularity at the right level of

abstraction. If the modules learn very high-level skills, then the modules will become task-specific, while if the modules learn very low-level skills, a large number of modules need to be trained.

2. Learning a diverse set of modules: Another big challenge is how to train the modules so that they learn *diverse* skills and do not *collapse* to the same skill.

Due to these limitations, several works use hand-crafted modules, i.e., they train the modules to learn specific skills. While this approach may work reasonably well for the single-task setup, scaling it to lifelong learning (or even multi-task) setups has been difficult.

## 5.2 Parameter Isolation Systems

*Parameter Isolation* based approaches work on the idea that different tasks should have their own set of “isolated” parameters. If no two tasks share any parameters, training on any task can not cause catastrophic forgetting on the other task. The idea of “isolating parameters” is in direct contrast to the idea of “learning composable modules” as the parameter isolation approaches focus more on avoiding catastrophic forgetting. In contrast, the modular network-based approaches focus more on knowledge transfer (both forward and backward). Even though these approaches are designed to avoid catastrophic forgetting by not sharing parameters, catastrophic forgetting happens because parameters are updated using loss from multiple tasks (and not because parameters are used to make predictions on several tasks). In practice, this means that parameters of different tasks can be used together in the forward pass but not in the backward pass, thus enabling forward transfer of knowledge, but not a backward transfer of knowledge.

However, in practice, many parameter isolation-based approaches can be described as modular neural networks with slightly different inductive biases. For example, a *controller-like* mechanism can be used to select which parameters belong to which task. Despite their similarity (and close relationship with) modular architectures, we consider them as a different category because:

1. Unlike the modular systems, which explicitly share modules across many tasks (thereby enabling both forward and backward knowledge transfer), parameter isolation-based approaches use different parameters for each task.
2. Parameter isolation approaches focus on avoiding catastrophic forgetting (at the expense of backward transfer).
3. Often, these approaches add more parameters as new tasks are encountered. Hence the number of parameters grows as  $\mathcal{O}(n)$  where  $n$  is the number of tasks. Thus one frequently encountered challenge is to reduce the memory footprint when adding new tasks. On the other hand, the common challenge for modular networks is to ensure that the knowledge is decomposed into different skills.

Parameter Isolation Systems are generally studied across two dimensions: (i) fixed capacity networks and (ii) increasing capacity networks.

### 5.2.1 Fixed Capacity Networks

The first sub-category of parameter isolation networks consists of models with a fixed capacity (i.e., the model’s capacity does not change as it trains over subsequent tasks). The isolated set of parameters is instantiated by learning task-specific *masks* which are used to determine which parameters will be used in the forward/backward pass for which task. These approaches trade-off catastrophic forgetting with model capacity by controlling the gradient flow through the network. Different mask-based approaches differ in terms of (i) what is masked and (ii) how are the mask values computed.

**Masking-based Approaches.** In parameter isolation systems, it is not required that the parameters for a given task will always form a contiguous block of parameters (like a feedforward layer). The masking function can operate at the fine level of parameters, i.e., the masking function can generate a mask per parameter. The general idea of masking is based on previous works like Courbariaux *et al.* (2015) and Hubara

*et al.* (2016) that train neural networks with binary-valued weights. The basic idea is to use a masking function to binarize real-valued weights during the forward pass and update the real-valued weights using the gradients corresponding to the binarized weights. Other works (Guo *et al.*, 2016) have used implicit masking functions that use the magnitude of the weight as a criterion for masking.

Mallya *et al.* (2018) proposed to learn bit-wise binary masks for each task. These masks are used to activate/deactivate the weights of a fixed and shared backbone network (on which the masks are applied) by element-wise multiplication with the binary masks. During training, the first step is to pre-train the backbone network. Mallya *et al.* (2018) used the ImageNet dataset (Deng *et al.*, 2009) and reported that the pretrained backbone network works well across multiple tasks. When training on a given task, a mask network is used to generate real-valued weights. These weights are mapped to binary masks using a deterministic thresholding function. These masks are multiplied (elementwise) with the weights of the backbone network, to generate the task-specific weights. The entire setup is trained end to end based on ideas from network binarization (Courbariaux *et al.*, 2015; Courbariaux *et al.*, 2016) and pruning (Guo *et al.*, 2016). Once the model is trained on the given task, the masking network is discarded, and only the task-specific bitwise mask is retained. The learned masks “piggyback” on the backbone network to solve a given task. One limitation of this approach is the dependence on pretraining the backbone network as the randomly initialized backbone networks perform quite poorly in practice.

Learning binary bit-masks may appear too restrictive in practice, and some works have explored the possibility of learning real-valued masks as well, although learning real-valued masks can be difficult in practice due to the possibility of forgetting the knowledge useful for the previous tasks. As such, learning real-valued masks requires designing the system carefully.

Serra *et al.* (2018) extend the idea of learning task-specific binary masks over weights to task-specific *almost-binary* attention masks over features (or activations). The proposed method, HAT (Hard Attention to Tasks), uses attention vectors of the previous tasks to define a mask for the current task and constrain the updates to the model’s weights.



The paper motivates this approach as follows: *When training on a sequence of tasks, different tasks could reuse the same intermediate features, but in different ways. For example, given a dataset of birds and dogs, the first task may require the model to differentiate between birds and dogs, and the second task may require the model to differentiate between black and brown animals. In such cases, the task identifier could be a useful feature for the model to perform well on both tasks, by conditioning the network layers on the task identifier.* The per-layer attention weight is computed as follows:

$$\mathbf{a}_l^t = \sigma(s\mathbf{e}_l^t),$$

where  $\mathbf{a}_l^t$  is the attention vector for the  $l^{\text{th}}$  layer of the model and  $\mathbf{e}_l^t$  is the single-layer task embedding for the  $l^{\text{th}}$  layer of the model, when training on the  $t^{\text{th}}$  task.  $\sigma$  is a gating function which squashes any input to the range  $[0, 1]$  and acts as a *pseudo-step* function. A hard step function is not used to enable flow of gradients to the layers of the model.  $s$  denotes a positive scaling parameter that controls the *hardness* of the *pseudo-step* function. In the case of the final layer, the attention vector is binary-hardcoded (since the final layer is a task-conditioned multi-headed output layer). After training the model on  $t^{\text{th}}$  task, a cumulative attention mask is computed by taking an element-wise max on all the previous attention vectors. i.e.,  $\mathbf{a}_l^{\leq t} = \max(\mathbf{a}_l^t, \mathbf{a}_l^{\leq t-1})$ . Since max operation is used, any feature (that was important for any of the previous tasks) gets a high attention score. When computing the gradient on the  $t + 1^{\text{th}}$  task, the gradient  $g_{l,ij}$ , corresponding to the  $i^{\text{th}}$  output and  $j^{\text{th}}$  input units in the  $l^{\text{th}}$  layer is masked using  $[1 - \min(a_{l,i}^{\leq t}, a_{l-1,j}^{\leq t})]$ . This mask prevents large updates to weights that are important for the previous tasks. When computing the attention weights,  $s$  is annealed in an epoch as follows:

$$s = \frac{1}{s_{max}} + (s_{max} - \frac{1}{s_{max}}) \left( \frac{b-1}{B-1} \right),$$

where  $s_{max}$  is a large positive constant ( $\gg 1$ ),  $B$  is the total number of batches in an epoch and  $b$  is the number of batches seen so far in the epoch. So at the start of the epoch, all features are approximately equally likely to be activated, and as training progresses, the selection becomes

more binarized. However, the gradient annealing scheme introduces some optimization challenges. Specifically, the embeddings  $e_i^t$  do not change much during training, and gradient magnitude is weak. Hence an additional gradient compensation term is introduced to compensate the effects of the annealed sigmoid, and some intermediate values (like  $|se_i^t|$ ) are clamped to obtain well-behaved gradients. For the  $t^{\text{th}}$  task, activations with a hard attention value are dedicated to that task. A sparsity constraint is introduced to reserve some model capacity for subsequent tasks by constraining the capacity spent on each task. During inference,  $s$  is set to  $s_{max}$  so that the gating function behaves like a step function.

Generally, *masking approaches* uses the same set of masks for both forward and backward transfer. While this makes sense in practice (weights that were not used during the forward pass can not have gradients during the backward pass), it also reduces the model’s flexibility, in terms of transferring knowledge across the tasks.

Masana *et al.* (2020b) proposed using ternary, instead of binary, masks for each task. Similar to Serra *et al.* (2018), these masks are applied to the features (or activations) of each layer and not to the weights. Since the number of activations tends to be smaller than the number of weight parameters, the memory overhead of Masana *et al.* (2020b) is lower in practice, compensating for the need to store 2x bits per feature instead of 1 binary bit per weight. However, unlike Serra *et al.* (2018), the masks are binary (and not real-valued) as only binary masks can guarantee to avoid *forgetting* of previous tasks. Two sets of masks are learned - the masks for features that will be *used* (forward pass) and another set for the features that will be *learned* (backward pass). This is related to how freezing layers work, features used in the forward pass do not necessarily have to be used in the backward pass. Specifically, the features can be in one of the three states: *used* (forward pass only), *learned* (forward and backward pass), and *unused* (neither forward nor backward pass).

Generally, *masking approaches* do not allow any change to the features corresponding to the previous tasks, so Masana *et al.* (2020b) add task-specific feature normalization that makes the (previously learned) features more optimal use for the later tasks. However, feature

normalization comes at the price of storing two extra floating point numbers per activation per task.

**Pruning-Based Approaches.** An alternate design choice would be to iteratively free-up parameters (via, say, network pruning) so that the subsequent tasks can use the parameters freed by the previous tasks, without having to add new parameters. PackNet (Mallya and Lazebnik, 2018) uses network pruning approach from previous works like Han *et al.* (2015) and Han *et al.* (2016) and proposes to sequentially *pack* multiple tasks into a single network by performing iterative pruning and network re-training. One major difference between PackNet and previous masking-based works is, all the unmasked parameters (that is, parameters that have not already been masked in the previous tasks) are used during both forward and backward passes. The mask for the  $i^{th}$  task is computed after training is finished on the  $i^{th}$  task. The training setup is quite straightforward. The model starts by training on the first task. After convergence, a certain percentage of weights are *pruned* i.e., set to 0. The network is retrained on the current task (without using the pruned weights) to account for the effect of pruning. When the model is trained on the second task, the unpruned weights (used in the first task) are kept fixed, and the pruned weights (not used in the second task) are trained. After convergence, some of the weights (trained in the second task) are pruned, and the model is trained on the unpruned weights (from the second task). This process is repeated every time a new task is added. In each round of pruning, the weights (in all the convolutional and fully connected layers) are sorted by their absolute magnitude, and the lowest 50% (or 75%) weights are pruned. The paper reports that subsequent retraining uses half as many epochs as original training.

One important thing to note is that PackNet is evaluated in setups where up to three new tasks are added. While some of the previous works have used fewer tasks (Kirkpatrick *et al.*, 2017; Li and Hoiem, 2017), scaling PackNet to a large number of tasks may require some changes. For example, it is expected that at some point, the network’s capacity would have to be increased, probably using approaches like Net2Net (Chen *et al.*, 2015b; Sodhani *et al.*, 2020).

**Learning paths in the network.** Fernando *et al.* (2017) proposed PathNet that uses evolutionary algorithms to discover *paths* (subparts of the network) to re-use for new tasks. A PathNet is a deep neural network having  $L$  layers, with each layer consisting of  $M$  modules. A module is said to be *active* if it is on the currently selected path. At most,  $N$  (3/4 in practice) modules can be active in any layer at any time. A pathway is represented by a matrix of at most  $N \times L$  integers. The integers in the  $i^{th}$  column refer to the active modules of the  $i^{th}$  layer. The output of each layer is summed up before passing to the subsequent layers. Pathways can evolve in two ways - serially or parallelly. In the **Serial Pathway Evolution**  $P$ , pathways are initialized randomly and are represented by a matrix of at most  $N \times L$  integers. A binary tournament selection algorithm is used where two pathways are selected randomly and trained for  $T$  epochs. The *fitness* of a pathway is measured in terms of classification error during training. The winning path is mutated by randomly selecting some modules (on the path) and swapping them with nearby modules in the layer. In the **Parallel Pathway Evolution**, multiple paths are trained in parallel, and as some paths are trained, they are compared with the other trained paths. The winning path overrides the losing paths, followed by a round of mutations.

After the model has learned a task, the best pathway is fixed, and its parameters are no longer updated. Modules that are not part of the best path are reinitialized. As the model trains on the next task, the same procedure (sampling random paths, comparing paths, and mutating the winner) is repeated. The paper reports that the previous best paths are active (during forward pass) in the RL experiments but not in the supervised learning experiments. The previous best paths are never used in the backward pass.

One limitation of the work is that it is evaluated on a sequence of only two tasks, which could limit its usefulness in practice.

Related to previous works on selecting a task-specific *path* through the network, Rajasegaran *et al.* (2019b) proposed RPS-Net (Random Path Selection Network) that starts with some random candidate paths and discovers the optimal path for a given task. The network consists of  $L$  distinct layers, where each layer has a set of  $M$  modules, stacked in parallel, along with a skip connection. During training, path selection is

performed for every  $J$  task. During path selection,  $N$  paths are randomly chosen and followed by the training process. The best path is then used for the next  $J$  tasks. Since the previously selected paths are fixed, the computation remains bounded, as at most one module for each of the  $L$  layers is being trained. In practice, the work also leverages techniques from regularization (Chapter 3) in the form of knowledge distillation and experience replay (Chapter 4), thus making it a hybrid approach. Additionally, a controller is used to balance between the current task loss and the knowledge distillation loss. The controller increases the weight of the knowledge distillation loss as the training progresses.

One important strength of RPS-Net is that, during inference, it does not need to know the task to which the given data point belongs as a common inference path is used.

### 5.2.2 Expert-Based Systems

An extreme version of Parameter Isolation Systems is the idea of adding a new network/model per task (which may or may not use predictions from the previously trained models). A very popular instantiation of this approach is Progressive Neural Networks (Rusu *et al.*, 2016) where a new *column* (network) is added every time a new task is encountered. The newly added column has lateral connections to the previous tasks that enable the forward transfer of knowledge from the previously trained models to the newly added model. During training, only the newly added model is trained, and the old weights are kept fixed, thus protecting from catastrophic forgetting. The obvious downside of this approach is that the number of parameters increases linearly with the number of tasks. Progressive Neural Networks also notes that the newly added models are not used to their full capacity, thus leaving scope for improvement.

Schwarz *et al.* (2018) propose a related, but much more memory efficient idea where two networks are maintained. One network (referred to as the *active column*) is used for training on the current task, and the second network (referred to as the *knowledge base*) stores the knowledge for solving the previous tasks. Training happens in two phases. In the first phase (called *progress* phase), the *active column* is trained on the

current task. Once the *active column* converges, the second phase starts where the *active column* is distilled into the *knowledge base*. This phase is referred to as the *compress* phase. Thus training over multiple tasks proceeds as a sequence of *progress* and *compress* steps, and the approach is known as Progress and Compress. During the *progress* phase, lateral connections between the knowledge base and the active column as used to transfer knowledge from the previous tasks to the current task. Only the active column is trained during the progress phase. This is similar to how Rusu *et al.* (2016) works. However, unlike Rusu *et al.* (2016), extra care needs to be taken to avoid catastrophic forgetting during *compress* stage. To that end, the paper uses a modified version of Elastic Weight Consolidation (Lee *et al.*, 2017). Specifically, when *compressing* the knowledge of the  $k^{\text{th}}$  task into the knowledge base, the following loss is optimized, with respect to the parameters  $\theta^{KB}$  of the knowledge base (while keeping the parameters of the active column fixed):

$$\mathbb{E} \left[ KL(\pi_k(\cdot|x) \parallel \pi^{KB}(\cdot|x)) \right] + \frac{1}{2} \|\theta^{KB} - \theta_{k-1}^{KB}\|_{\gamma F_{k-1}^*}^2, \quad (5.4)$$

where  $\pi_k(\cdot|x)$  and  $\pi^{KB}(\cdot|x)$  are the outputs of the active column (after learning on  $k^{\text{th}}$  task) and knowledge base respectively.  $x$  represents the input,  $\mathbb{E}$  denotes the expectation over either the data under the active column,  $\theta_{k-1}^{KB}$  and  $F_{k-1}^*$  represent the mean and the diagonal Fisher of the online EWC Gaussian approximation resulting from previous tasks, and  $\gamma$  is a hyperparameter.

**Relation to Modular Neural Networks.** The general architecture of adding task-specific experts is similar to the work on modular neural networks. In fact, the high-level motivation is also very similar: when doing lifelong learning, different tasks share a common substructure or common sub-problems. We hope to capture some of these shared structures/knowledge via the experts, i.e., different experts will learn to solve different tasks, and this knowledge can be shared across tasks. In practice, mixture-of-expert systems generally do not *compose* modules for a given data point or use simplistic aggregation operations like *average* (similar to how ensembles work). The controller mechanism either selects  $k$  experts (or assigns soft-attention scores to all the experts).

The application setup is closer to *system identification or task identification* (Zadeh, 1956; Åström and Bohlin, 1965; Swevers *et al.*, 1997; Bhat *et al.*, 2002; Gevers *et al.*, 2006; Ljung, 2010; Van Overschee and De Moor, 2012; Chiuso and Pillonetto, 2019; Ajay *et al.*, 2019; Yu *et al.*, 2017; Zhu *et al.*, 2017) setup, and the goal is to identify the correct expert to use for a given task. In the mixture-of-experts-based setup, the controller mechanism is often non-parameterized, as we explain in the examples below.

Aljundi *et al.* (2017) proposed using a mixture of experts for the lifelong learning setup by training task-specific experts as follows: The model is initialized with one expert which is trained on the first task. As the model trains on subsequent tasks, new experts are added to the model and trained on the newly added tasks. This training setup does not require access to the previously seen data. Moreover, as new experts are added with new tasks, the model does not have the challenge of *capacity saturation*. There are two challenges that need to be addressed: (i) how to select the *correct* expert during inference and (ii) adding one new expert per task leads to linear growth in the number of parameters as new tasks are encountered.

The first challenge is addressed as follows: a set of gating auto-encoders whose job is to decide which expert should be used for a given sample. Specifically, each expert uses a shallow auto-encoder (which is trained with the expert). During inference, all the auto-encoders encode the input sample, and the expert, corresponding to the auto-encoder with the smallest reconstruction error, is selected. The use of auto-encoders also provides a mechanism to measure the *relatedness* between the different tasks, by comparing the reconstruction error between the different encoders. Specifically, given the  $i^{th}$  and the  $j^{th}$  tasks, the *relatedness* is given as:

$$Rel(T_i, T_j) = 1 - \frac{Er_j - Er_i}{Er_i}, \quad (5.5)$$

where  $Er_i$  is the reconstruction error for the  $i^{th}$  task. This task relatedness is used to select the most related task (for a new task) to be used as a prior model for learning the new task. Note that in the general neural modular networks, this problem is solved by the controller

module and the mechanism used by Aljundi *et al.* (2017) (or in general mixture-of-expert based approaches) can be seen as a non-parametric controller.

The second challenge, of the linear growth of the number of model parameters with new tasks, violates one of the desiderata of lifelong learning systems 2.6: the number of parameters should increase sub-linearly with the number of tasks. Aljundi *et al.* (2017) leaves this limitation for the future work.

Rannen *et al.* (2017) use mixture-of-experts based architecture by training task-specific auto-encoders for mitigating catastrophic forgetting. Their proposed solution works as follows: Let us say that the model has a shared feature extractor, a shared model trunk, and some task-specific layers (that use the activations from the trunk as the input). After training the model on the first task, an auto-encoder is trained on the representations from the first task, in order to capture the most important features from the first task. Specifically, the auto-encoder is trained with two losses: (i) the reconstruction loss and (ii) the supervised learning loss using the data from the first task (i.e., the features learned by the auto-encoder should be informative enough to solve the first task). When the model is trained on the second task, two constraints are added, along with the supervised learning loss on the second task. The first constraint is in the form of distillation loss used in Li and Hoiem (2017). As described in Section 3.4, the distillation loss aims to mitigate the influence of the use of different data distributions. The second constraint ensures that the features learned by the auto-encoder (for the first task) are still good for performance on the first task. This procedure can be applied to a sequence of tasks. Like Aljundi *et al.* (2017), this method also leads to linear growth in the number of parameters. Rannen *et al.* (2017) justify the trade-off by arguing that the memory footprint of the encoders is much smaller than the memory footprint of the overall model.

### 5.2.3 Expanding Networks

An important aspect of lifelong learning that does not get as much focus is Capacity Saturation. Only a few selective works have focused on that



problem (Yoon *et al.*, 2018; Sodhani *et al.*, 2020). Specifically, Yoon *et al.* (2018) proposed Dynamically Expandable Network (DEN), that can increase the network capacity dynamically as it trains on a sequence of tasks. There are three key steps involved in training DEN: Selective retraining, Dynamic network expansion, and Network split/duplication.

**Selective retraining.** At the start, the network is trained with L1 regularization to induce sparsity. As new tasks are encountered, a sparse linear model is trained to solve the task, using the topmost hidden units of the network. The nodes/weights, that changed in the top layer, provide the starting positions for breadth-first search, to find the nodes (in the layers below) that have paths to the nodes in the topmost layer. Only the weights of the selected sub-network are trained.

**Dynamic Network Expansion.** If selective retraining is not sufficient to train the model on a given task, its capacity is increased by expansion. This step is made efficient by using group sparse regularization. Specifically, the capacity of each layer is increased by  $k$  neurons. Group sparsity regularization removes the hidden units that are not necessary for training, thus preventing the wasteful addition of neurons as new tasks are encountered. This is one significant improvement over previous approaches like Progressive Neural Networks (Rusu *et al.*, 2016).

The last key step is **Network Split**. As the model trains through multiple tasks, the semantic drift of the neurons is tracked, and if the semantic drift becomes too high, the neuron is split into two copies. After the split/duplication step, the network is trained again (to ensure task knowledge is not lost).

Some recent works have looked into hand-designing experts based on the task/domain setup. Li *et al.* (2020) proposed to leverage compositionality to develop a new approach for lifelong learning in sequence-to-sequence tasks. They build on the idea of decomposing syntactic and semantic representations with compositionality by learning separate representations for the semantic and syntactic knowledge (Li *et al.*, 2019b) and the networks encoding these two representations are analogous to experts in the mixture-of-expert based systems. There is no explicit gating mechanism, and the representation from the modules is used for making the final prediction. The network encoding the syntax is trained

only on the first task and not updated afterward (with the assumption that the syntax does not change across the tasks).

Chen *et al.* (2015b) proposed network expansion techniques that can grow a smaller network into a larger network using function-preserving transformations. The paper presents two variants: (i) Net2WiderNet (for expanding the width of a given network) and (ii) Net2DeeperNet (for expanding the depth of a given network).

Consider a neural network where the  $i^{\text{th}}$  and the  $i + 1^{\text{th}}$  layers are fully connected layers and layer  $i$  uses a point-wise non-linearity. If the  $i^{\text{th}}$  layer has  $m$  inputs and  $n$  outputs and the  $i + 1^{\text{th}}$  layer has  $p$  outputs, then the Net2WiderNet operation can be used to widen  $i^{\text{th}}$  layer by replacing it with a layer that has  $N$  outputs, where  $N > n$ . First, a random mapping function  $r$ , from  $\{1, 2, \dots, N\} \rightarrow \{1, 2, \dots, n\}$ , is defined as follows:

$$r(j) = \begin{cases} j & j \leq n \\ \text{random sample from } \{1, 2, \dots, n\} & j > n \end{cases}$$

The weight matrices  $W^i$  and  $W^{i+1}$  are replaced by  $U^i$  and  $U^{i+1}$  such that

$$U_{k,j}^{(i)} = W_{k,g(j)}^{(i)},$$

$$U_{j,h}^{(i+1)} = \frac{1}{|\{x | g(x) = g(j)\}|} W_{g(j),h}^{(i+1)}.$$

The first  $n$  columns of  $W^i$  are copied as it is into  $U^i$ . Columns  $n + 1$  through  $N$  of  $U^{(i)}$  are created by randomly selecting columns of  $W$  (with replacement), followed by normalization with a replication factor (computed using the frequency of sampled columns).

Similarly, the Net2DeeperNet operation can be used to replace the  $i^{\text{th}}$  layer (representing  $\phi(h^{(i-1)\top} W^{(i)})$ ) by a deeper layer (representing  $\phi(U^{(i)\top} \phi(W^{(i)\top} h^{(i-1)}))$ ). The newly added  $U$  matrix is initialized as identity matrix and updated during finetuning. One limitation of this approach is that  $\phi$  (the activation function) must satisfy  $\phi(I\phi(v)) = \phi(v)$  for all vectors  $v$  where  $I$  is an identity matrix. So while this operation holds for activations like ReLU, using this operation with maxout units requires some modifications to the  $U$  matrix and the operation does not hold for the sigmoid activation.

The technique is proposed in the context of accelerating the training of a large neural network by first training a smaller network and then expanding it into a larger network. However, the paper mentions lifelong learning as an application area for their approach. Indeed, subsequent works have applied the technique for alleviating capacity saturation. For example, Sodhani *et al.* (2018) combined Net2Net (specifically Net2WiderNet) with Gradient Episodic Memory to create a hybrid model. When training on a given task, the model uses Gradient Episodic Memory updates to retain the knowledge of the previous task. When the model's capacity is saturated (and the model is unable to learn new tasks), it is expanded using the Net2Net approach, and the enlarged model continues to train on the subsequent tasks. Sodhani *et al.* (2018) shows that the combined approach works better in practice than the individual components.

### 5.3 Summary

In this chapter, we introduced the architecture-based methods proposed for lifelong learning that assign a model copy to every new task that arrives. We started by discussing the idea of using Modular networks, their motivation from multiple perspectives, their architectures, and finally, their benefits to lifelong learning. Understanding a more general biologically-inspired type of method, called modular networks, is necessary because they provide the ability to use different model architectures as constituent modules, add new modules throughout training, and also help in representing composable skills.

Parameter isolation methods are the types of architecture-based methods which are similar to modular networks. The main difference is that parameter isolation methods have a different inductive bias such that the tasks have their own set of isolated parameters. They are further divided into fixed capacity networks and increasing capacity networks. Fixed capacity networks can be masking-based, pruning-based, and involve discovering paths in the network as different modules. There are expert-based methods that require adding a new network for each task. This is similar to modular networks in the sense that the task-specific experts are continuously added to the main network.

Finally, we presented the capacity saturation aspect of lifelong learning by describing the DEN method. This method avoids capacity saturation and ensures stable learning for future tasks by following three important steps: selective retraining, dynamic network expansion, and network split/duplication.

# 6

---

## Benchmarks

---

Lifelong learning methods today are designed and implemented with different assumptions and setups. In earlier chapters, we described how the lifelong learning methods can be categorized from an algorithmic point of view. But benchmarking the proposed methods in different datasets with different settings makes comparing these approaches very complicated. Such untrustworthy benchmarks and comparisons raise the importance of introducing some systematic benchmarks in lifelong learning. With the availability of huge datasets in the machine learning domain, there is an abundance of benchmarks to evaluate lifelong learning methods. In this chapter, we go over some of the most commonly used benchmarks by dividing them based on their application: vision-based and NLP-based.

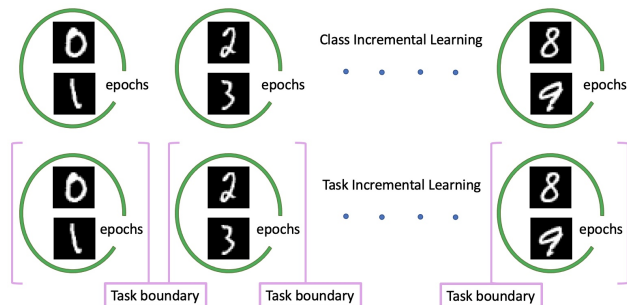
### 6.1 Vision Benchmarks

Many methods proposed in lifelong learning are evaluated on image-based benchmarks. These benchmarks are typically adapted from fields such as image classification, reinforcement learning (Atari games, robot manipulation, imitation), and generative models (Lesort *et al.*, 2019). To evaluate a lifelong learning method, these benchmarks are modified,

augmented, and concatenated together to create sequences of tasks. In this section, we describe such benchmarks in detail and also describe the benchmarks that are designed specifically for lifelong learning settings.

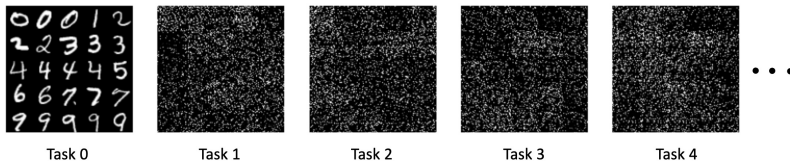
### 6.1.1 Variants of existing datasets

**MNIST:** Early lifelong learning methods, with the focus on vision tasks, started with benchmarking methods on some variation of the MNIST dataset (LeCun *et al.*, 2010). Permuted MNIST (Goodfellow *et al.*, 2015) and Split MNIST (Lopez-Paz and Ranzato, 2017b) were some of the early benchmarks for lifelong learning. Figure 6.1 illustrates a simple training protocol on the Split MNIST benchmark. It shows both class and task incremental learning cycle. MNIST dataset contains training samples for supervised learning of classification of handwritten digits zero to nine. In Split MNIST benchmark, the tasks are basically the disjoint sets of classes from the MNIST dataset. The model has to learn from the training samples that arrive sequentially from these tasks at each training step. Following the supervised learning, the model iterates over given training samples for several epochs and is evaluated on all classes seen so far. In the Split MNIST benchmark (Zenke *et al.*, 2017; Farquhar and Gal, 2019; Aljundi *et al.*, 2019b; Ven and Tolia, 2019a; Swaroop *et al.*, 2019; Ven and Tolia, 2019b), the difference between task and class incremental learning is the awareness of the model to the task shift. To this end, in the task incremental learning, trained on Split MNIST, the model knows the tasks' boundaries. Permuted



**Figure 6.1:** Class and task incremental learning on the Split MNIST benchmark.

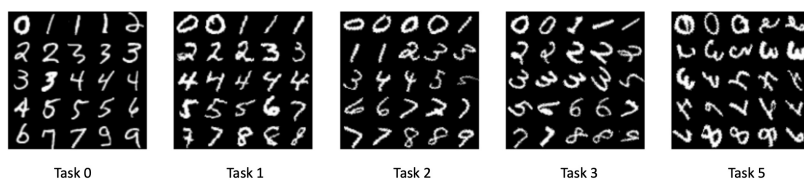
MNIST is another benchmark that was introduced after the Split MNIST benchmark. The approach of creating the Permuted MNIST benchmark is straightforward. In this case, the model receives all training samples of ten digits at each training time. The model learns from a regular MNIST dataset as the first task. Then, the model receives the permuted version of regular MNIST as the second task. So the model should learn from the permuted image sample and also should not forget what it learned at the previous task. Similarly, in the next steps, the model will receive samples that have different permutations and will have to adapt without forgetting catastrophically. Figure 6.2 shows the flow of training samples that the model should learn overtime in the Permuted MNIST benchmark.



**Figure 6.2:** Permuted MNIST that have been used as a benchmark for assessing the lifelong learning methods.

Lopez-Paz and Ranzato (2017b) introduced and used the Rotated MNIST benchmark to evaluate their Gradient of Episodic Memory (GEM) method. Rotated MNIST is a more practical and meaningful benchmark than the Permuted MNIST. In this benchmark, the goal of the model is to have a robust behavior through time by learning from a sequence of tasks that differ by rotation transformation. In other words, the images in each task are MNIST images with some fixed degree rotation transformation. Figure 6.3 gives an example of using Rotated MNIST in a lifelong learning setup.

KMNIST is a dataset, adapted from Kuzushiji Dataset that consists of Kuzushiji-MNIST, Kuzushiji-49, and Kuzushiji-Kanji datasets (Clanuwat *et al.*, 2018). KMNIST Dataset is a drop-in replacement for the MNIST dataset. KMNIST chooses one character to represent each of the 10 rows of Hiragana. KMNIST adds some complexity to the Split MNIST benchmark since the shape of characters is more complex than the shape of simple digits.



**Figure 6.3:** Rotated MNIST benchmark. Each block shows few samples that the model should learn at each training step.



**Figure 6.4:** Split KMNIST sample that can be used in a lifelong learning task.

Figure 6.5 shows the combination of three datasets: MNIST, FashionMNIST (dataset of the articles of clothing at low resolution (Xiao *et al.*, 2017)), and KMNIST datasets in form of a sequence of three tasks also known as the MNIST Fellowship benchmark. In this benchmark, each task is a variation of the MNIST dataset that arrives in a sequence. MNIST Fellowship has 30 classes in total and each instance is in the size of 28x28 images.



**Figure 6.5:** MNIST Fellowship benchmark. Each task could be created using a subset of either MNIST, Fashion MNIST, or KMNIST datasets.

**ILSVRC2012 and CIFAR:** As explained above Split MNIST, Permuted MNIST, KMNIST, and MNIST Fellowship benchmarks are the simple benchmarks that have been used in early research in lifelong learning. As the field grew, having a more complex benchmark became crucial for evaluating lifelong learning methods. To create more complex and difficult benchmarks split versions of other popular datasets like CIFAR and ILSVRC2012 have been used. The perfor-



mance of the models trained on lifelong learning settings are mostly reported for MNIST (LeCun *et al.*, 2010), Permuted MNIST, rotated MNIST, CIFAR-10, CIFAR-100 (Krizhevsky, Hinton, *et al.*, 2009), ImageNet (Deng *et al.*, 2009) where data is split into sequences of classes or tasks. MNIST, Permuted MNIST, and rotated MNIST are usually split into two and five consecutive sets of classes (Zenke *et al.*, 2017; Kirkpatrick *et al.*, 2017; Lopez-Paz and Ranzato, 2017b; Zenke *et al.*, 2017; Nguyen *et al.*, 2017; Lesort *et al.*, 2018). CIFAR-100 is split into two, five, ten, or twenty sets of classes such that the model should learn each set of classes at each time consecutively (Lopez-Paz and Ranzato, 2017b; Zenke *et al.*, 2017; Rebuffi *et al.*, 2017; Hou *et al.*, 2019; Castro *et al.*, 2018). Recently, most of the approaches report the performance of models on split ImageNet (Rebuffi *et al.*, 2017; Hou *et al.*, 2019; Castro *et al.*, 2018; Wu *et al.*, 2019b) and Celeb-10000 (Wu *et al.*, 2019b). Some approaches benchmark on datasets such as ImageNet, CIFAR-100, SVHN, UCF101, Omniglot, GTSR, DPed, Flower, Aircraft, and DTD (Li *et al.*, 2019a).

**5-datasets** : Recently proposed methods benchmark using several existing datasets as a sequence of individual tasks. For instance, **5-datasets** is a sequence of five different datasets as five 10-way classification tasks (Serra *et al.*, 2018; Saha *et al.*, 2021; Ebrahimi *et al.*, 2020). These datasets are: **CIFAR-10**, **MNIST**, **SVHN** (Netzer *et al.*, 2011), **notMNIST** (Bulatov, 2011), and **Fashion-MNIST** (Xiao *et al.*, 2017).

### 6.1.2 CORE50

Lomonaco and Maltoni (2017) introduced CORE50 that is constructed specifically to evaluate lifelong learning methods. Continual object Recognition benchmark (known as CORE50) consists of 50 domestic objects belonging to 10 categories of simple objects including plug adapters, mobile phones, scissors, light bulbs, cans, glasses, balls, markers, cups, and remote controls. Since the lifelong learning setup pose significant challenges for deep learning models, having clean object-centered instances may reduce learning complexity at each training step. With this

idea, Lomonaco and Maltoni (2017) attempted to collect clean images centered by the main object and avoid having other objects in the same image like in ImageNet.

Figure 6.6 shows some examples of CORE50 training samples. As shown in Figure 6.6 objects are presented in each instance such that the camera point-of-view mimics the operator’s eyes point of view. In this way, models can learn a training sample that is simple and provided to the model clearly with minimum complexity. The original benchmark was designed for studying lifelong learning in the robotic domain and it contained short videos instead of images. In order to collect samples, the operator smoothly moves his/her arm to present objects from different angles. It is worth mentioning that the operator changes hands throughout the sessions.

This provides a chance to produce more samples for relevant objects. Lomonaco and Maltoni (2017) collected data in 11 distinct sessions that



**Figure 6.6:** Some examples of CORE50 training samples. The grabbing hand (left or right) changes while collecting images of different objects.

included samples from eight indoor and three outdoor views designated by different backgrounds and lighting. Each object in the corresponding session presented by a 15 seconds video (at 20 fps) has been recorded with a Kinect 2.0 sensor delivering 300 RGB-D frames.

For low-data streams on lifelong learning setup, Antoniou *et al.* (2020) introduced a benchmark that defines a systematic approach for the setting of continual few-shot learning on various datasets such that the performance of the model is reported on each dataset individually. The learning agent has access to a very limited set of training samples in most real-world scenarios in each task as described by Antoniou *et al.* (2020). However, most proposed approaches in lifelong learning need to

revisit training samples for several epochs at each training step. That could be considered as the first limitation of the above benchmarks. To this end, benchmarking the lifelong learning methods by using the introduced benchmarks in this chapter such that visiting the training samples are allowed once or for a few numbers of epochs (Hayes *et al.*, 2020; Laleh *et al.*, 2020) are interesting and challenging approaches to evaluating lifelong learning methods.

The second limitation of the current approaches is that they alleviate catastrophic forgetting through time for incrementally learned classes or tasks that arise from the same distribution and same dataset. To this end, current approaches have been focused on a homogeneous lifelong learning problem. To make the benchmark more realistic and closer to real-life scenarios some other benchmarks have been proposed recently to include more challenges for assessing lifelong learning methods. In the next section, we introduce some of the recently proposed benchmarks that try to overcome one or both limitations explained above.

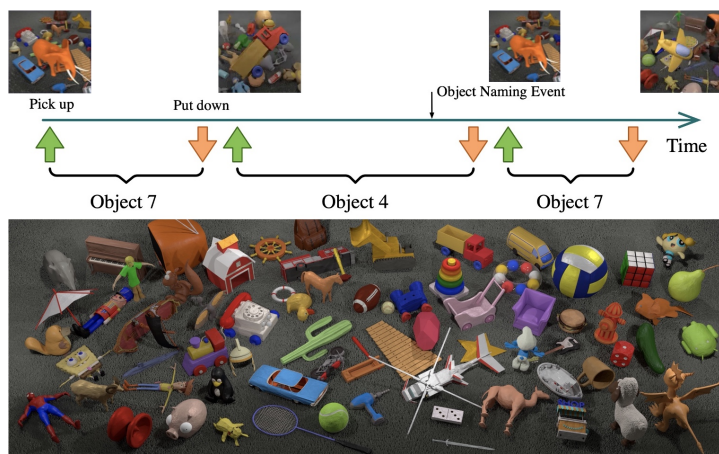
### 6.1.3 CRIB

Stojanov *et al.* (2019) introduced CRIB benchmark that is motivated by infant learning concepts in psychology. Infant learning is characterized based on five pillars: Incremental learning, Repetitive exposure to objects, Temporally contiguous visual experience, self-supervised learning (since labeling events are sparse and noisy), and Object Instance learning that precedes categorization. The infant object learning process consists of repetitive exposure to the object, starting with toys in their environment, which is an inherently incremental procedure. The visual experience that infants can see is temporary continuous and poorly smooth, particularly for the first two years. Their supervision is fairly sparse and they do not have constant object naming supervision by their parent. Infants tend to pick up different toys while playing, explore them for a few minutes, and then put them down. This is a continuous pattern observed in infants.

Therefore, they may pick up a bunch of different objects that only get naming supervision from their parents for a small subset of them. Stojanov *et al.* (2019) tried to connect incremental learning with infant

learning by having a simple model of object interaction by picking up and putting down objects during episodes. They called picking an object up and examining it for a while, then putting it down and picking another as one learning exposure.

The object can be presented to the learner after a while again. But if it has been seen before, the label will not be provided to the learner. That can mimic the sparse supervision situation. To model such a procedure, they used Toys-200 3D that contains object samples in the dataset with a toy-like appearance. They generate a small smooth video of the picked-up object and rotate it in front of a background in various lighting conditions and then put it down and select another toy object. The selected object in each training sample will not appear in the background. Figure 6.7 illustrates the CRIB benchmark idea and the exposure learning process in the lifelong learning approach.



**Figure 6.7:** CRIB benchmark. Toys-200 dataset of 200 unique toy objects that model infants learning in a lifelong learning approach.

#### 6.1.4 OpenLORIS-Object

OpenLORIS is another well-known benchmark that simulates closely a real-life scenario for a lifelong learning agent (She *et al.*, 2020). (L)ifel(O)ng (R)obotic V(IS)ion (OpenLORIS) is an Object Recognition benchmark that is designed for facilitating lifelong learning research

primarily for the robotic domain and extended to other application domains as well. This benchmark examines the capability of learning the common objects in the home scenario in some limited conditions. Since fully retraining robotic agents at each time for a new task or same task but with different lighting in the environment or different point of view is infeasible, lifelong learning methods can help to overcome such challenges but alleviate catastrophic forgetting and evaluating proposed methods in such conditions need benchmarks that can assess the agents' capabilities in these conditions. She *et al.* (2020) include the common challenges that the learning agents might need to deal with in the environment such as changes in illumination, occlusion, object size, camera-object distance, camera-object angle, and clutter. Including these factors from real-life environments into the benchmark creates more realistic scenarios for assessing a lifelong learning agent's capabilities.

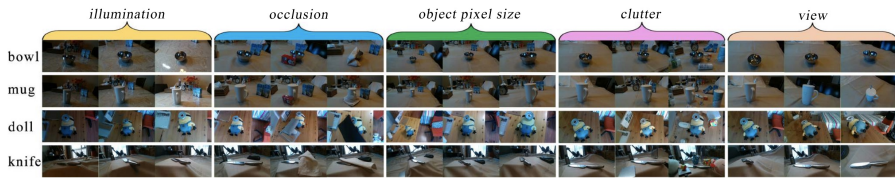


Figure 6.8: OpenLORIS-Object benchmark.

### 6.1.5 Stream-51

Conventional neural networks require running a loop over a batch of i.i.d data multiple times to improve performance. Although several lifelong learning training methods are proposed to alleviate forgetting under non-i.i.d. conditions, many of these methods are inflexible and inefficient in real-world scenarios where data comes from a dynamic data distribution and constantly changes over time.

Stream-51 is used to evaluate whether an agent can robustly handle shifts and novel inputs in training data in a lifelong learning scenario. Online streaming learning is a more pragmatic approach where a model needs to learn one sample at a time that it receives from a stream

of data. Stream-51 provides sufficient data classes with high-quality training instances for such an online lifelong learning setup.

This benchmark consists of temporally correlated images from 51 unique object categories and additional evaluation classes for testing novelty recognition. The evaluation samples are not provided to the model at training time. As Figure 6.9 shows, the model learns from a temporally correlated stream of samples. Similar to the other benchmarks, the model is evaluated on the previously seen classes and the ability of the model to detect unlearned concepts.

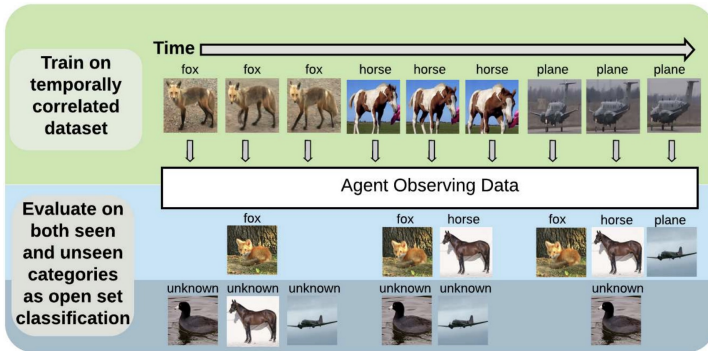


Figure 6.9: Stream-51

**Stream-51 Specific Metrics:** In the Stream-51 benchmark 6.1.5, the model receives samples in a stream of data. A few of these samples are considered unseen data or novel samples. Therefore the goal is to compute the overall classification performance and its ability to detect novel inputs (Roady *et al.*, 2020). Since Stream-51 works in the stream of data where the model is not allowed to iterate over the training samples for several epochs, the overall classification performance is computed as follows:

$$\Omega_{\text{Classif.}} = \min \left( 1, \frac{1}{T} \sum_{t=1}^T \frac{\alpha_t}{\alpha_{\text{offline},t}} \right), \quad (6.1)$$

where  $T$  is the total number of testing events,  $\alpha_t$  is the accuracy of the streaming learner at time  $t$ , and  $\alpha_{\text{offline},t}$  is the accuracy of an optimized offline model at time  $t$ . This metric normalizes a streaming

learner’s performance using an optimized offline learner. Normalizing the streaming learner’s performance to an offline learner makes the metric easier to interpret across various orderings (Roady *et al.*, 2020). For novelty detection, Roady *et al.* (2020) propose an incremental variant of the area under the OSC curve (AUOSC) which normalizes an incremental learner’s performance to an optimized offline baseline as follows:

$$\Omega_{\text{AUOSC}} = \min \left( 1, \frac{1}{T} \sum_{t=1}^T \frac{\gamma_t}{\gamma_{\text{offline},t}} \right), \quad (6.2)$$

where  $T$  is the total number of testing events,  $\gamma_t$  is the AUOSC score of the incremental learner at time  $t$ , and  $\gamma_{\text{offline},t}$  is the AUOSC score of the optimized offline learner at time  $t$  (Roady *et al.*, 2020).

### 6.1.6 IIRC

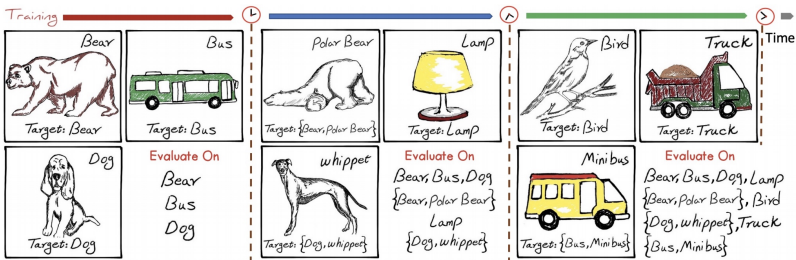
Abdelsalam *et al.* (2021) introduced Incremental Implicitly-Refined Classification (IIRC) as an extension to the class incremental learning setup. Unlike other lifelong learning benchmarks, IIRC breaks the assumption of having the same level of hierarchy for the samples the model should learn from through time. IIRC provides a benchmark and scenario which is more challenging and more aligned with real-life learning scenarios.

In this setup, the incoming batches of classes might have two levels of granularity: a high level (coarse) label like “bear” and a low-level (fine) label like “polar bear”. Only one label is provided at a time, and the model has to figure out the other label if it has already learned it. Therefore, the model should be able to expand its knowledge about a concept while not forgetting high-level information that is previously learned knowledge (at a different granularity) about the same concept. Another challenge included in the IIRC benchmark is that the classes have imbalanced sample distribution similar to real-life where not all classes are observed at the same frequency.

IIRC benchmark provides IIRC-ImageNet and IIRC-CIFAR built based on ImageNet and CIFAR datasets respectively. These two datasets are most popular in lifelong learning literature. IIRC-ImageNet simulates

data diversity challenges. They provided a shorter version of IIRC-ImageNet called IIRC-ImageNet-li. Abdelsalam *et al.* (2021) clearly states that the IIRC-ImageNet-lite version is not for benchmarking the model performance but only for performing and debugging experiments.

IIRC benchmark reveals the model’s ability to expand its knowledge and associate and re-associate labels over time. Figure 6.10 illustrates the training and evaluation paradigm in the IIRC benchmark. In the figure, the top right label is available to the label model during training, whereas the bottom label, defined as a target, is predicted by the model during evaluation. The right bottom panel also shows the set of classes used for model evaluation, and the dashed line represents the task boundary in the task incremental learning setup.



**Figure 6.10:** The learning and evaluation procedure in the IIRC benchmark.

### IIRC Specific Metrics:

As discussed previously, the Average Accuracy metric is one of the most popular metrics used in lifelong learning. But multi-label classification setup that involves hierarchy requires a different evaluation metric. IIRC, a benchmark closer to real-life scenarios, enabled the multi-labels classification for super-classes and sub-classes that the model might see over its lifetime. In the IIRC benchmark, limiting the model to not predicting a large number of possible classes is essential, otherwise the model will tend to predict more labels to receive more positive feedback during its supervision periods.

Sorower (2010) proposes to use Exact-Match Ratio (MR) as a metric for the multi-label classification. The Exact-Match Ratio (MR) is defined



and computed as:

$$MR = \frac{1}{n} \sum_{i=1}^n I(Y_i == \hat{Y}_i), \quad (6.3)$$

where  $I$  is the indicator function,  $Y_i$  are the ground truth labels for sample  $i$ ,  $\hat{Y}_i$  is the set of predictions for the corresponding sample, and  $n$  is the total number of samples. The Exact-Match Ratio penalizes partial correct predictions in the same way as completely incorrect ones. In other words, a partially corrected prediction will have the same score as the completely incorrect one. That is an important weakness of the Exact-Match Ratio metric.

In partial multi-labels or complete multi-label classification literature using Jaccard Similarity (JS) (Sorower, 2010) is very common. The Jaccard Similarity (JS) computes the score for evaluating a model’s performance for the multi-labels prediction as intersection over the union of true labels. It is mathematically formalized as follows:

$$JS = \frac{1}{n} \sum_{i=1}^n \frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|}. \quad (6.4)$$

The precision weighted JS (pw-JS) is further weights JS by sampling precision. Similar to the other lifelong learning benchmarks, after training the model on the task  $j$ , pw-JS is computed to evaluate the model performance on all tasks from the beginning to task  $j$  as follow:

$$R_j = \frac{1}{n} \sum_{i=1}^n \frac{|Y_i \cap \hat{Y}_i|}{|Y_i \cup \hat{Y}_i|} \times \frac{|Y_i \cap \hat{Y}_i|}{|\hat{Y}_i|}, \quad (6.5)$$

where  $n$  is the total number of evaluation samples for all the tasks seen so far,  $Y_i$  is the set of ground truth labels of sample  $i$  and  $\hat{Y}_i$  is the set of predictions from the model for sample  $i$  (Abdelsalam *et al.*, 2021). Compared to JS, the weights additionally differentiate completely correct predictions from partially correct predictions. Abdelsalam *et al.* (2021) show that algorithms that tend to generate more labels have lower pw-JS scores than JS scores.

## 6.2 NLP Benchmarks

Natural language processing is used in many day-to-day applications that take the advantage of ML to solve tasks. With the surge of attention on using lifelong learning methods in practical scenarios, having a good benchmark to evaluate proposed methods is a key element in applying the lifelong learning methods in the NLP domain. Several benchmarks have been proposed recently in this direction and we discuss some of them in this section.

### 6.2.1 Personalized Online Language Learning and FIREHOSE datasets

Hu *et al.* (2020) collected data from the popular social media platform Twitter to introduce a benchmark for evaluating a personalized language model for each user. Each user is considered as a different task in this setup. The goal here is to propose a Personalized Online Language Learning (POLL) method that helps in finding a personalized language model for each user over time. The key characteristic of this setup is that users are added and dropped over time. Since users tweet with different frequencies, the setup has a highly non-stationary distribution. This setup provides both multi-task and continual settings of tasks. Learning the language model for each user is an online multi-task setting since each user is considered a task in this setup. Since it requires learning a shared language model with the non-stationary data distribution, it is a lifelong learning problem (Hu *et al.*, 2020).

Hu *et al.* (2020) collected more than 100 million tweets with more than 1.5 billion tokens, posted by one million users over six calendar years called the FIREHOSE datasets. Since language learning is an unsupervised or semi-supervised problem in real life, this dataset does not require human or automatic labeling process to create the benchmark for evaluating lifelong learning algorithms. Therefore, FIREHOSE is a massive web-scale dataset that can also support research on POLL. Hu *et al.* (2020) provided a smaller version of the dataset called Firehose10M and the bigger one as Firehose100M. Table 6.1 provides the statistic for each version.

**Table 6.1:** The statistic for both Firehose10M and Firehose100M (Hu *et al.*, 2020).

	# Users	# Tweets	# Tokens
Firehose10M	94.0K	10.4M	173.3M
Firehose100M	917.4K	100.4M	1672.7M

### 6.2.2 Text Classification and Question Answering Datasets

Text classification tasks have a lot of practical and industrial usage. Therefore, having a lifelong learning benchmark to evaluate proposed method’s performance in continual text classification tasks seems quite important. Masson d’Autume *et al.* (2019) used the publicly available text classification datasets that are collected from various domains such as news classification, sentiment analysis, Wikipedia article classification, and questions and answers categorization. They constructed a benchmark using AGNews (4 classes), Yelp (5 classes), DBPedia (14 classes), Amazon (5 classes), and Yahoo (10 classes) datasets. Table 6.2 shows the benchmark specification in detail.

**Table 6.2:** The datasets that are used in a lifelong learning Benchmark proposed to evaluate methods on text classification and question answering for consecutive tasks (Masson d’Autume *et al.*, 2019).

Dataset	Task type	# Classes
AGNews	news classification	4
Yelp	sentiment analysis	5
Amazon	sentiment analysis	5
DBPedia	Wikipedia article classification	14
Yahoo	questions and answers categorization	10

In Table 6.2, both Yelp and Amazon dataset are used for the same purpose (sentiment classification). Therefore the selected classes are merged into the same semantic analysis task. Masson d’Autume *et al.* (2019) provided two versions for benchmarking lifelong learning methods: a balanced dataset that contains 115,000 training and 7,600 test examples and an imbalanced version that contains 575,000 training and 38,000 test examples. The second version is imbalanced due to the

**Table 6.3:** The datasets that are used in a challenging lifelong learning benchmark proposed to evaluate methods on text classification (Mehta *et al.*, 2021). All tasks are either single sentence or sentence pair classification. # Train, # Dev, # Test denotes the number of examples in train, valid, test splits respectively. # L denotes the number of classes for each tasks.

Dataset	Task	Source Domains(s)	# Train	# Valid	# Test	# L	Metrics
CoLA (Warstadt <i>et al.</i> , 2019)	Linguistic Acceptability	Journal articles and books	7,695	856	1,043	2	Matthews correlation
BoolQ (Clark <i>et al.</i> , 2019)	Boolean Question Answering	Google queries, Wikipedia passages	8,483	944	3,270	2	Acc.
SST-2 (Socher <i>et al.</i> , 2013)	Sentiment Analysis	Movie reviews	9,971	873	872	2	Acc.
QQP (Wang <i>et al.</i> , 2018)	Paraphrase Detection	Quora questions	10,794	4,044	4,043	2	Acc. & F1
YahooQA (Zhang <i>et al.</i> , 2015)	Q & A Categories	Yahoo! Answers	13,950	4,998	4,998	10	Acc.
Yelp (Zhang <i>et al.</i> , 2015)	Review Rating Prediction	Business reviews	12,920	3,999	3,998	5	Acc.
Decomp (Poliak <i>et al.</i> , 2018)	Event Factuality	FactBank	10,176	4,034	3,934	2	Acc.
AAC (Stab <i>et al.</i> , 2018)	Argument Aspect Detection	Web documents	10,893	2,025	4,980	3	Acc. & F1
DISCONN8 (Prasad <i>et al.</i> , 2019; Kim <i>et al.</i> , 2020)	Explicit Discourse Marker Prediction	Penn Discourse TreeBank	9,647	1,020	868	8	Acc. & F1
QNLI (Wang <i>et al.</i> , 2018)	Question Answering NLI	Wikipedia	9,927	5,464	5,463	2	Acc.
RocBSO (Mostafazadeh <i>et al.</i> , 2016)	Binary Sentence Order Prediction	Roc story corpus	10,000	2,400	2,400	2	Acc.
MNLI (Williams <i>et al.</i> , 2018)	Natural Language Inference	speech, fiction, govt. reports	11,636	4,816	4,815	3	Acc.
SciTAIL (Khot <i>et al.</i> , 2018)	Multi-choice Science QA	Science exams	11,145	1,305	1,304	2	Acc.
PDTB2L1 (Prasad <i>et al.</i> , 2019; Kim <i>et al.</i> , 2020)	Implicit Discourse Relation Classification	Penn Discourse TreeBank	13,046	1,183	1,046	4	Acc. & F1
Emotion (Saravia <i>et al.</i> , 2018)	Emotion Detection	Twitter	9,600	2,000	2,000	6	Acc. & F1

different number of examples selected from each source. To study lifelong learning under the realistic scenario of a large number of tasks, Mehta *et al.* (2021) introduces a novel suite of 15 diverse text classification tasks. They showed that the introduced suite proves more challenging than the previously discussed benchmark with 5 datasets. Table 6.3 shows the benchmark datasets in detail. Future works should consider repurposing ExMix (Aribandi *et al.*, 2022), a massive collection of 107 supervised tasks across diverse domains to be an even more challenging and realistic lifelong language learning benchmark. To construct a lifelong learning benchmark for the question and answering tasks, Masson d’Autume *et al.* (2019) create their benchmark using the dataset described in Table 6.4.

**Table 6.4:** The datasets that are used in a lifelong learning Benchmark for consecutive question-answering tasks (Masson d’Autume *et al.*, 2019).

Dataset	Characteristics	Source	# Training	# Test
SQuAD 1.1 (Rajpurkar <i>et al.</i> , 2016)	Reading comprehension	Wikipedia articles	90,000	10,000
TriviaQA (Joshi <i>et al.</i> , 2017)	QA pairs written by trivia enthusiasts and evidence	Wikipedia	60,000	8,000
TriviaQA (Joshi <i>et al.</i> , 2017)	QA pairs written by trivia enthusiasts and evidence	Web	76,000	10,000 (unverified)
QuAC (Choi <i>et al.</i> , 2018)	Information-seeking dialog-style	Wikipedia article and a teacher answers with a short excerpt from the article	80,000	7,000

### 6.2.3 Word and Sentence Representations

Word embedding plays a huge role in improving NLP task performance. Some NLP applications may need access to distributed word vectors that can be built sequentially. In a large general-purpose corpus, finding the embedding may not be useful for domain-specific downstream tasks because languages and the meaning of vocabulary can slightly change over time. Emerging social media cause a faster change in the meaning of some vocabularies. For example, the meanings of a word used in specific situations and contexts may differ in some viral videos or posts. As a result, Biesialska *et al.* (2020) argue that learning word embedding poses an important lifelong learning paradigm. A lifelong learning setup can fulfill the requirements of vocabulary changes over time. According to our best knowledge, there is still a lack of a good benchmark

for evaluating lifelong learning methods on learning embedding in a lifelong learning setup.

### 6.2.4 Dialogue Systems

Lifelong learning in task-oriented dialogue systems has been previously studied by Lee (2017), who perform lifelong learning over three tasks in an end-to-end (E2E) dialogue modeling setting. Wu *et al.* (2019a) propose a dialogue state tracking (DST) model that trains on a set of domains and can transfer knowledge to a new domain without forgetting. They evaluate DST on different domains in the MultiWOZ dataset (Budzianowski *et al.*, 2020). Mi *et al.* (2020) propose a method for lifelong learning in the natural language generation (NLG) setting, where different domains from the MultiWOZ dataset are presented in a sequence. Geng *et al.* (2021) also propose a lifelong learning method in the NLG setting that uses network pruning and expansion to adapt to new tasks. They evaluate on a benchmark of 7 tasks, composed of domains from the In-Car Assistant (Eric and Manning, 2017), MultiWOZ, and CamRest (Wen *et al.*, 2017) datasets. However, these methods look at lifelong learning in task-oriented dialogue with a relatively small number of tasks (3-7), with a smaller amount of data, and in a single setting.

Madotto *et al.* (2020) propose a benchmark that presents 37 tasks in a sequence in 4 different settings. They consider single-domain dialogues from 4 different datasets: Taskmaster-1 and Taskmaster-2 (Byrne *et al.*, 2019), Schema-Guided Dialogue (SGD) dataset (Rastogi *et al.*, 2020), and MultiWOZ (Budzianowski *et al.*, 2020). Each domain in the datasets forms a single task in the lifelong learning formulation. The four settings they consider are as follows:

- Intent prediction (INTENT), where an intent label has to be predicted from the given conversation history.
- Dialogue state tracking (DST), where the intent and a sequence of slot-value pairs that are passed to an API call to be tracked have to be predicted from the conversation history.
- Natural language generation (NLG), where a natural language response has to be predicted given the result of an API call.
- End-to-end (E2E), which combines the three settings mentioned above.

Madotto *et al.* (2020) unify the settings described above into a sequence-to-sequence learning problem by presenting the dialogue history as a sequence of turns and serializing the API calls.

**Metrics on Lifelong Dialogue Systems:** To evaluate the dialogue systems, Madotto *et al.* (2020) adopt a modular approach for evaluating the models in each setting with a suitable metric as follows:

**Table 6.5:** The statistics of four tasks that are collected to create a benchmark for Dialog systems in lifelong learning setup (Madotto *et al.*, 2020).

Name	#Samples			#Domains	#Intents	Average #turns
	Train	Valid	Test			
TM19	4,403	551	553	6	112	19.97
TM20	13,839	1,731	1,734	7	128	16.92
MWoZ	7,906	1,000	1,000	5	15	13.93
SGD	5,278	761	1,531	19	43	14.71
Total	31,426	4,043	4,818	37	280	16.23

- *INTENT*: Intent accuracy, which compares the predicted intent label against the gold intent label.
- *DST*: Joint goal accuracy (JGA) metric, which measure the proportion of samples for which all slot-value pairs are predicted correctly.
- *NLG*: Entity error rate (EER) which measures the number of slots in the input that do not appear in the generated utterance, and the BLEU score between the generated and gold utterance.
- *E2E*: All the above metrics.

To measure performance in the lifelong learning setting, the average value of the metric over the tasks is used. Forward or backward transfer are measured by aggregating the metrics defined above.

### 6.3 Summary

To achieve state-of-the-art performance in real-world scenarios, lifelong learning methods must be capable of generalization on a broad range of complex datasets comprising a large number of tasks. The goal of generalization requires lifelong learning benchmarks to be challenging and closer to real-world applications to ensure robust behavior over time.

We first described numerous vision-based benchmarks involving multiple tasks of image classification with varying levels of difficulty by inducing shifts in the data distribution. We also described NLP domain-based benchmarks. These benchmarks are designed specifically for the type of task at hand such as text classification, question answering, text representation, dialogue systems, etc.

Some of these benchmarks also required introducing new metrics for evaluations for a more intuitive evaluation of the baselines. It is worth noting

that several benchmarks for lifelong learning presented in this chapter are inspired by multi-task learning benchmarks and are often used without any modification.



# 7

---

## Future Challenges

---

So far, we have looked at both a high-level overview of lifelong learning systems (definition and desiderata of lifelong learning systems and the relationship between lifelong learning and other paradigms) as well as specific instances of lifelong learning algorithms and benchmarks and metrics for evaluating lifelong learning systems. In this last section, we conclude the primer with a discussion on future challenges and important research directions in lifelong learning systems.

The first important research direction is **developing methods for organizing (or compartmentalizing) the system’s knowledge**. This would enable the system to manipulate specific parts of its knowledge without overwriting all the previous knowledge. Such methods would be useful in the context of catastrophic forgetting and faster adaptation to new tasks as the system can choose which knowledge it wants to forget/update and retain all the other knowledge. These methods are especially relevant in the context of training large-scale models on constantly evolving datasets (like the web data where the ground truth data is continuously changing). Ability to selectively update parts of the knowledge system is a crucial requirement for deploying such systems in the real world (Bommasani *et al.*, 2021).

One promising line of work in this direction is the work on *adapters* (Houlsby *et al.*, 2019; Pfeiffer *et al.*, 2021; Pfeiffer *et al.*, 2020) - small modules that are added to the intermediate layers of a large, pretrained transformer model. When transferring to a downstream task, only the newly added adapter modules are finetuned, and the original, pretrained transformer model is kept fixed.

Previous works like Houlsby *et al.* (2019), Pfeiffer *et al.* (2020), and Pfeiffer *et al.* (2021) have demonstrated that finetuning just the adapter modules can also provide excellent performance on the downstream tasks. The use of adapters enables learning downstream models that are *compact* (uses fewer trainable parameters for each downstream task) and *extensible* (uses new adapters that can be incrementally trained on downstream tasks). Adapters can be seen as *knowledge modules* that can be swapped in and out, depending on the downstream task. However, adapters are designed for the forward transfer of knowledge (start with a pre-trained model and adapt to the new tasks) and do not provide a straightforward mechanism to enable the backward transfer of knowledge (updating the knowledge in the pre-trained model, in a non-catastrophic way, while training on the new tasks). Combining insights from existing work in modular architectures (Chapter 5) and adapters could provide a useful inductive bias for compartmentalizing the systems knowledge in terms of its parameters.

Another important research direction is to **enable the lifelong learning systems to interface with external knowledge sources**. This capability is especially relevant in the context of interactive systems (like dialogue systems or embodied agents) operating in the real world. These interactive systems are often likely to encounter new topics/objects/items and have to account for the continually changing state of the world. For example, it is not possible for a static language model, trained at one point in time, to answer “what is the current temperature in Montreal” without access to an external knowledge base that can provide it the current temperature in Montreal. One way to work around this challenge is to develop lifelong learning systems that learns *skills* (e.g., querying the external knowledge-bases, aggregating the results from different queries, etc.) instead of knowledge. One popular approach for teaching such *skills* is to curate a list of tasks, with or without a curriculum, such that each task requires the system to either learn a new skill or learn to combine previously learned skills (Bengio *et al.*, 2009; Chevalier-Boisvert *et al.*, 2019; Weston *et al.*, 2015). For example, a question-answering system may start by learning to choose relevant facts from a curated list of facts. The system may learn to create a curated list of facts using a single knowledge base as the next step. In the third task, the system may have to answer the question using the knowledge base (requiring it first to create a list of facts and then answer the question). In the next task, the system could learn to choose the relevant knowledge base(s) to query. In this way, the system learns a sequence of skills and combines them for solving new tasks. The key challenge with this approach is that the list of tasks is often hand-designed, thus limiting the scope, both in terms of the number of tasks and the spectrum of skills they cover.

We note that the system doesn’t need to rely exclusively on external

knowledge bases for all the information. The system could employ memory-augmented neural networks (like Neural Turing Machine (Graves *et al.*, 2014; Lüders *et al.*, 2016), Differentiable Neural Computer (Graves *et al.*, 2016) or memory network (Sukhbaatar *et al.*, 2015; Sukhbaatar *et al.*, 2015; Kumar *et al.*, 2016; Miller *et al.*, 2016; Lample *et al.*, 2019)). The key requirement is that the system should be able to query external knowledge stores for accessing the dynamically changing information. This research direction can be seen as an extension to the first direction where the compartmentalized knowledge lives in external, well-curated knowledge bases that the system can access on-demand.

## References

---

- Abdelsalam, M., M. Faramarzi, S. Sodhani, and S. Chandar. (2021). “IIRC: Incremental Implicitly-Refined Classification”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11038–11047.
- Agarwal, A., A. Rakhlin, and P. Bartlett. (2008). “Matrix regularization techniques for online multitask learning”. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-138*.
- Ahn, H., S. Cha, D. Lee, and T. Moon. (2019). “Uncertainty-based continual learning with adaptive regularization”. *arXiv preprint arXiv:1905.11614*.
- Ajay, A., M. Bauza, J. Wu, N. Fazeli, J. B. Tenenbaum, A. Rodriguez, and L. P. Kaelbling. (2019). “Combining physical simulators and object-based networks for control”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 3217–3223.
- Alet, F., M. Bauza, A. Rodriguez, T. Lozano-Perez, and L. P. Kaelbling. (2018a). “Modular meta-learning in abstract graph networks for combinatorial generalization”. *arXiv preprint arXiv:1812.07768*.
- Alet, F., T. Lozano-Pérez, and L. P. Kaelbling. (2018b). “Modular meta-learning”. *arXiv preprint arXiv:1806.10166*.
- Alex Davies, A. M. (2021). “Guide to Self Driving Cars”. *The Wired*. Sept. URL: <https://www.wired.com/story/guide-self-driving-cars> (accessed on 09/09/2021).

- Aljundi, R., F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars. (2018). “Memory aware synapses: Learning what (not) to forget”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 139–154.
- Aljundi, R., E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia. (2019a). “Online Continual Learning with Maximal Interfered Retrieval”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2019/file/15825aee15eb335cc13f9b559f166ee8-Paper.pdf>.
- Aljundi, R., L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars. (2019b). “Online Continual Learning with Maximally Interfered Retrieval”. arXiv: [1908.04742](https://arxiv.org/abs/1908.04742) [cs.LG].
- Aljundi, R., P. Chakravarty, and T. Tuytelaars. (2017). “Expert gate: Lifelong learning with a network of experts”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3366–3375.
- Aljundi, R., K. Kelchtermans, and T. Tuytelaars. (2019c). “Task-Free Continual Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Aljundi, R., M. Lin, B. Goujaud, and Y. Bengio. (2019d). “Gradient based sample selection for online continual learning”. In: *Advances in Neural Information Processing Systems*. 11816–11825.
- Amer, M. and T. Maul. (2019). “A review of modularization techniques in artificial neural networks”. *Artificial Intelligence Review*. 52(1): 527–561.
- Andreas, J., M. Rohrbach, T. Darrell, and D. Klein. (2016a). “Learning to compose neural networks for question answering”. *arXiv preprint arXiv:1601.01705*.
- Andreas, J., M. Rohrbach, T. Darrell, and D. Klein. (2016b). “Neural module networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 39–48.
- Andrychowicz, M., F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. (2017). “Hindsight experience replay”. *arXiv preprint arXiv:1707.01495*.

- Antoniou, A., M. Patacchiola, M. Ochal, and A. Storkey. (2020). “Defining Benchmarks for Continual Few-Shot Learning”. arXiv: 2004.11967 [cs.CV].
- Aribandi, V., Y. Tay, T. Schuster, J. Rao, H. S. Zheng, S. V. Mehta, H. Zhuang, V. Q. Tran, D. Bahri, J. Ni, J. Gupta, K. Hui, S. Ruder, and D. Metzler. (2022). “ExT5: Towards Extreme Multi-Task Scaling for Transfer Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Vzh1BFUCiIX>.
- Åström, K. J. and T. Bohlin. (1965). “Numerical Identification of Linear Dynamic Systems from Normal Operating Records”. eng. In: *Proc. IFAC Conference on Self-Adaptive Control Systems*.
- Atherton, L., D. Dupret, and J. Mellor. (2015). “Memory trace replay: the shaping of memory consolidation by neuromodulation.” *Trends in Neurosciences*. 38(9): 560–570.
- Auda, G. and M. Kamel. (1998). “Modular neural network classifiers: A comparative study”. *Journal of Intelligent and Robotic Systems*. 21(2): 117–129.
- Auda, G. and M. Kamel. (1999). “Modular neural networks: a survey”. *International Journal of Neural Systems*. 9(02): 129–151.
- Badia, A. P., B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskiy, Z. D. Guo, and C. Blundell. (2020). “Agent57: Outperforming the atari human benchmark”. In: *International Conference on Machine Learning*. PMLR. 507–517.
- Baevski, A., H. Zhou, A. Mohamed, and M. Auli. (2020). “wav2vec 2.0: A framework for self-supervised learning of speech representations”. *arXiv preprint arXiv:2006.11477*.
- Bahdanau, D., K. Cho, and Y. Bengio. (2014). “Neural machine translation by jointly learning to align and translate”. *arXiv preprint arXiv:1409.0473*.
- Bapst, V., T. Keck, A. Grabska-Barwińska, C. Donner, E. D. Cubuk, S. S. Schoenholz, A. Obika, A. W. Nelson, T. Back, D. Hassabis, et al. (2020). “Unveiling the predictive power of static structure in glassy systems”. *Nature Physics*. 16(4): 448–454.

- Bassett, D. S., D. L. Greenfield, A. Meyer-Lindenberg, D. R. Weinberger, S. W. Moore, and E. T. Bullmore. (2010). “Efficient physical embedding of topologically complex information processing networks in brains and computer circuits”. *PLoS comput biol.* 6(4): e1000748.
- Bellman, R. (1978). “An introduction to artificial intelligence: can computer think?” *Tech. rep.*
- Bengio, S., Y. Bengio, J. Cloutier, and J. Gessei. (2013). “On the optimization of a synaptic learning rule”. In: *Optimality in Biological and Artificial Networks?* Routledge. 281–303.
- Bengio, Y. (2012). “Deep learning of representations for unsupervised and transfer learning”. In: *Proceedings of ICML workshop on unsupervised and transfer learning.* JMLR Workshop and Conference Proceedings. 17–36.
- Bengio, Y., T. Deleu, N. Rahaman, R. Ke, S. Lachapelle, O. Bilaniuk, A. Goyal, and C. Pal. (2019). “A meta-transfer objective for learning to disentangle causal mechanisms”. *arXiv preprint arXiv:1901.10912.*
- Bengio, Y., J. Louradour, R. Collobert, and J. Weston. (2009). “Curriculum learning”. In: *Proceedings of the 26th annual international conference on machine learning.* 41–48.
- Benzing, F. (2021). “Unifying Regularisation Methods for Continual Learning”. arXiv: [2006.06357 \[cs.LG\]](https://arxiv.org/abs/2006.06357).
- Bertolero, M. A., B. T. T. Yeo, and M. D’Esposito. (2015). “The modular and integrative functional architecture of the human brain”. *Proceedings of the National Academy of Sciences.* 112(49): E6798–E6807. ISSN: 0027-8424. DOI: [10.1073/pnas.1510619112](https://doi.org/10.1073/pnas.1510619112). eprint: <https://www.pnas.org/content/112/49/E6798.full.pdf>. URL: <https://www.pnas.org/content/112/49/E6798>.
- Bhat, K. S., S. M. Seitz, J. Popović, and P. K. Khosla. (2002). “Computing the physical parameters of rigid-body motion from video”. In: *European Conference on Computer Vision.* Springer. 551–565.
- Bhatt, H. S., R. Singh, M. Vatsa, and N. Ratha. (2012). “Matching cross-resolution face images using co-transfer learning”. In: *2012 19th IEEE International Conference on Image Processing.* 1453–1456. DOI: [10.1109/ICIP.2012.6467144](https://doi.org/10.1109/ICIP.2012.6467144).

- Bhuvaneshwari, V., M. Priyadharshini, C. Deepa, D. Balaji, L. Rajeshkumar, and M. Ramesh. (2021). “Deep learning for material synthesis and manufacturing systems: a review”. *Materials Today: Proceedings*.
- Bickel, S., C. Sawade, and T. Scheffer. (2008). “Transfer learning by distribution matching for targeted advertising”. In: *Proceedings of the 21st International Conference on Neural Information Processing Systems*. 145–152.
- Biesialska, M., K. Biesialska, and M. R. Costa-jussà. (2020). “Continual Lifelong Learning in Natural Language Processing: A Survey”. *Proceedings of the 28th International Conference on Computational Linguistics*. DOI: [10.18653/v1/2020.coling-main.574](https://doi.org/10.18653/v1/2020.coling-main.574). URL: <http://dx.doi.org/10.18653/v1/2020.coling-main.574>.
- Bommasani, R., D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.* (2021). “On the opportunities and risks of foundation models”. *arXiv preprint arXiv:2108.07258*.
- Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.* (2020). “Language models are few-shot learners”. *arXiv preprint arXiv:2005.14165*.
- Budzianowski, P., T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić. (2020). “MultiWOZ – A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling”. arXiv: [1810.00278](https://arxiv.org/abs/1810.00278) [cs.CL].
- Bulatov, Y. (2011). “Notmnist dataset”. *Tech. rep.* Google (Books/OCR). URL: <http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html>.
- Bullmore, E. and O. Sporns. (2009). “Complex brain networks: graph theoretical analysis of structural and functional systems”. *Nature reviews neuroscience*. 10(3): 186–198.
- Buzzega, P., M. Boschini, A. Porrello, and S. Calderara. (2021). “Re-thinking Experience Replay: a Bag of Tricks for Continual Learning”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2180–2187.
- Byrne, B., K. Krishnamoorthi, C. Sankar, A. Neelakantan, D. Duckworth, S. Yavuz, B. Goodrich, A. Dubey, A. Cedilnik, and K.-Y. Kim. (2019). “Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset”. arXiv: [1909.05358](https://arxiv.org/abs/1909.05358) [cs.CL].



- Caelli, T., L. Guan, and W. Wen. (1999). “Modularity in neural computing”. *Proceedings of the IEEE*. 87(9): 1497–1518.
- Carey, S. (1985). *Conceptual change in childhood*. MIT press.
- Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. (2010). “Toward an architecture for never-ending language learning.” In: *AAAI*. Vol. 5. Atlanta. 3.
- Caruana, R. (1997). “Multitask learning”. *Machine learning*. 28(1): 41–75.
- Castro, F. M., M. J. Marín-Jimenez, N. Guil, C. Schmid, and K. Alahari. (2018). “End-to-End Incremental Learning”. arXiv: [1807.09536](https://arxiv.org/abs/1807.09536) [cs.CV].
- Chang, M., A. Gupta, S. Levine, and T. L. Griffiths. (2019). “Automatically Composing Representation Transformations as a Means for Generalization”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=B1ffQnRcKX>.
- Chaudhry, A., P. K. Dokania, T. Ajanthan, and P. H. Torr. (2018). “Riemannian walk for incremental learning: Understanding forgetting and intransigence”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 532–547.
- Chaudhry, A., A. Gordo, P. Dokania, P. Torr, and D. Lopez-Paz. (2021). “Using Hindsight to Anchor Past Knowledge in Continual Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 6993–7001.
- Chaudhry, A., M. Ranzato, M. Rohrbach, and M. Elhoseiny. (2019a). “Efficient Lifelong Learning with A-GEM”. In: *International Conference on Learning Representations*.
- Chaudhry, A., M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. (2019b). “On tiny episodic memories in continual learning”. *arXiv preprint arXiv:1902.10486*.
- Chen, C., A. Seff, A. Kornhauser, and J. Xiao. (2015a). “Deepdriving: Learning affordance for direct perception in autonomous driving”. In: *Proceedings of the IEEE international conference on computer vision*. 2722–2730.
- Chen, T., I. Goodfellow, and J. Shlens. (2015b). “Net2net: Accelerating learning via knowledge transfer”. *arXiv preprint arXiv:1511.05641*.

- Chen, Z. J., Y. He, P. Rosa-Neto, J. Germann, and A. C. Evans. (2008). “Revealing modular architecture of human brain structural networks by using cortical thickness from MRI”. *Cerebral cortex*. 18(10): 2374–2381.
- Chen, Z., V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. (2018). “Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks”. In: *International Conference on Machine Learning*. PMLR. 794–803.
- Chen, Z., N. Ma, and B. Liu. (2015c). “Lifelong Learning for Sentiment Classification”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 750–756.
- Cheng, S. and L. M. Frank. (2008). “New Experiences Enhance Coordinated Neural Activity in the Hippocampus”. *Neuron*. 57(2): 303–313.
- Chevalier-Boisvert, M., D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. (2019). “BabyAI: First Steps Towards Grounded Language Learning With a Human In the Loop”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=rJeXCo0cYX>.
- Chiuso, A. and G. Pillonetto. (2019). “System identification: A machine learning perspective”. *Annual Review of Control, Robotics, and Autonomous Systems*. 2: 281–304.
- Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. (2014). “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. *arXiv preprint arXiv:1406.1078*.
- Choi, E., H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. (2018). “QuAC : Question Answering in Context”. arXiv: 1808.07036 [cs.CL].
- Chrysakis, A. and M.-F. Moens. (2020). “Online Continual Learning from Imbalanced Data”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. *Proceedings of Machine Learning Research*. PMLR. 1952–1961. URL: <http://proceedings.mlr.press/v119/chrysakis20a.html>.

- Ciresan, D. C., A. Giusti, L. M. Gambardella, and J. Schmidhuber. (2013). “Mitosis detection in breast cancer histology images with deep neural networks”. In: *International conference on medical image computing and computer-assisted intervention*. Springer. 411–418.
- Clanuwat, T., M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. (2018). “Deep learning for classical japanese literature”. *arXiv preprint arXiv:1812.01718*.
- Clark, C., K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. (2019). “BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2924–2936.
- Clune, J., J.-B. Mouret, and H. Lipson. (2013). “The evolutionary origins of modularity”. *Proceedings of the Royal Society b: Biological sciences*. 280(1755): 20122863.
- Cobbe, K., C. Hesse, J. Hilton, and J. Schulman. (2020). “Leveraging procedural generation to benchmark reinforcement learning”. In: *International conference on machine learning*. PMLR. 2048–2056.
- Courbariaux, M., Y. Bengio, and J.-P. David. (2015). “Binaryconnect: Training deep neural networks with binary weights during propagations”. *arXiv preprint arXiv:1511.00363*.
- Courbariaux, M., I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. (2016). “Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1”. *arXiv preprint arXiv:1602.02830*.
- Dai, W., O. Jin, G.-R. Xue, Q. Yang, and Y. Yu. (2009). “Eigentransfer: a unified framework for transfer learning”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. 193–200.
- De Lange, M., R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. (2019). “Continual learning: A comparative study on how to defy forgetting in classification tasks”. *arXiv preprint arXiv:1909.08383*. 2(6).
- Dekel, O., P. M. Long, and Y. Singer. (2006). “Online multitask learning”. In: *International Conference on Computational Learning Theory*. Springer. 453–467.

- Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. (2009). “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 248–255.
- Devin, C., A. Gupta, T. Darrell, P. Abbeel, and S. Levine. (2017). “Learning modular neural network policies for multi-task and multi-robot transfer”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2169–2176.
- Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova. (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. *arXiv preprint arXiv:1810.04805*.
- Dhar, P., R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa. (2019). “Learning without memorizing”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5138–5146.
- Dong, Q., S. Gong, and X. Zhu. (2017). “Multi-task curriculum transfer deep learning of clothing attributes”. In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 520–529.
- Douillard, A., Y. Chen, A. Dapogny, and M. Cord. (2021). “Plop: Learning without forgetting for continual semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4040–4050.
- Du, Y., W. M. Czarnecki, S. M. Jayakumar, R. Pascanu, and B. Lakshminarayanan. (2018). “Adapting auxiliary losses using gradient similarity”. *arXiv preprint arXiv:1812.02224*.
- Ebrahimi, S., M. Elhoseiny, T. Darrell, and M. Rohrbach. (2019). “Uncertainty-guided continual learning with bayesian neural networks”. *arXiv preprint arXiv:1906.02425*.
- Ebrahimi, S., F. Meier, R. Calandra, T. Darrell, and M. Rohrbach. (2020). “Adversarial Continual Learning”. arXiv: [2003.09553](https://arxiv.org/abs/2003.09553) [cs.LG].
- Elman, J. L. (1993). “Learning and development in neural networks: The importance of starting small”. *Cognition*. 48(1): 71–99.
- Eric, M. and C. D. Manning. (2017). “Key-Value Retrieval Networks for Task-Oriented Dialogue”. arXiv: [1705.05414](https://arxiv.org/abs/1705.05414) [cs.CL].

- Fan, H., B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer. (2021). “Multiscale vision transformers”. *arXiv preprint arXiv:2104.11227*.
- Farajtabar, M., N. Azizan, A. Mott, and A. Li. (2020). “Orthogonal gradient descent for continual learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 3762–3773.
- Farquhar, S. and Y. Gal. (2019). “Towards Robust Evaluations of Continual Learning”. arXiv: [1805.09733 \[stat.ML\]](https://arxiv.org/abs/1805.09733).
- Fernando, C., D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. (2017). “Pathnet: Evolution channels gradient descent in super neural networks”. *arXiv preprint arXiv:1701.08734*.
- Fini, E., S. Lathuiliere, E. Sangineto, M. Nabi, and E. Ricci. (2020). “Online continual learning under extreme memory constraints”. In: *European Conference on Computer Vision*. Springer. 720–735.
- Finn, C., P. Abbeel, and S. Levine. (2017). “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *ICML*.
- Ford, M. (2018). *Architects of Intelligence: The truth about AI from the people building it*. Packt Publishing Ltd.
- Foret, P., A. Kleiner, H. Mobahi, and B. Neyshabur. (2021). “Sharpness-aware Minimization for Efficiently Improving Generalization”. In: *International Conference on Learning Representations*.
- Frans, K., J. Ho, X. Chen, P. Abbeel, and J. Schulman. (2017). “Meta Learning Shared Hierarchies”. *arXiv e-prints*. Oct. arXiv: [1710.09767](https://arxiv.org/abs/1710.09767).
- French, R. M. (1999). “Catastrophic forgetting in connectionist networks”. *Trends in cognitive sciences*. 3(4): 128–135.
- Friston, K., J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny. (2007). “Variational free energy and the Laplace approximation”. *Neuroimage*. 34(1): 220–234.

- Ge, L., J. Gao, and A. Zhang. (2013). “OMS-TL: A Framework of Online Multiple Source Transfer Learning”. In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management. CIKM '13*. San Francisco, California, USA: Association for Computing Machinery. 2423–2428. ISBN: 9781450322638. DOI: [10.1145/2505515.2505603](https://doi.org/10.1145/2505515.2505603). URL: <https://doi.org/10.1145/2505515.2505603>.
- Gemp, I., B. McWilliams, C. Vernade, and T. Graepel. (2021). “EigenGame: {PCA} as a Nash Equilibrium”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=NzTU59SYbNq>.
- Geng, B., F. Yuan, Q. Xu, Y. Shen, R. Xu, and M. Yang. (2021). “Continual Learning for Task-oriented Dialogue System with Iterative Network Pruning, Expanding and Masking”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics. 517–523. DOI: [10.18653/v1/2021.acl-short.66](https://doi.org/10.18653/v1/2021.acl-short.66). URL: <https://aclanthology.org/2021.acl-short.66>.
- Gevers, M. *et al.* (2006). “System Identification without Lennart Ljung: what would have been different?” *Forever Ljung in System Identification, Studentlitteratur AB, Norrtälje*. 2.
- Girshick, R., J. Donahue, T. Darrell, and J. Malik. (2014). “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.
- Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. (2020). “Generative adversarial networks”. *Communications of the ACM*. 63(11): 139–144.
- Goodfellow, I. J., M. Mirza, D. Xiao, A. Courville, and Y. Bengio. (2015). “An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks”. arXiv: [1312.6211](https://arxiv.org/abs/1312.6211) [stat.ML].
- Gopnik, A. (1988). “Conceptual and semantic development as theory change: The case of object permanence”. *Mind & Language*. 3(3): 197–216.

- Gou, J., B. Yu, S. J. Maybank, and D. Tao. (2021). “Knowledge distillation: A survey”. *International Journal of Computer Vision*. 129(6): 1789–1819.
- Goyal, A., A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. (2021). “Recurrent Independent Mechanisms”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=mLcmdlEUxy->.
- Goyal, A., S. Sodhani, J. Binas, X. B. Peng, S. Levine, and Y. Bengio. (2020). “Reinforcement Learning with Competitive Ensembles of Information-Constrained Primitives”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=ryxgJTEYDr>.
- Graves, A., G. Wayne, and I. Danihelka. (2014). “Neural turing machines”. *arXiv preprint arXiv:1410.5401*.
- Graves, A., G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. (2016). “Hybrid computing using a neural network with dynamic external memory”. *Nature*. 538(7626): 471–476. ISSN: 00280836. URL: <http://dx.doi.org/10.1038/nature20101>.
- Gugerty, L. (2006). “Newell and Simon’s logic theorist: Historical background and impact on cognitive modeling”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 50. No. 9. SAGE Publications Sage CA: Los Angeles, CA. 880–884.
- Guo, Y., A. Yao, and Y. Chen. (2016). “Dynamic network surgery for efficient DNNs”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 1387–1395.
- Guo, Y., M. Liu, T. Yang, and T. Rosing. (2020). “Improved Schemes for Episodic Memory-based Lifelong Learning”. *Advances in Neural Information Processing Systems*. 33.
- Gupta, A. S., M. A. van der Meer, D. S. Touretzky, and A. D. Redish. (2010). “Hippocampal Replay Is Not a Simple Function of Experience”. *Neuron*. 65(5): 695–705.
- Gupta, G., K. Yadav, and L. Paull. (2020). “La-maml: Look-ahead meta learning for continual learning”. *arXiv preprint arXiv:2007.13904*.

- Hacohen, G. and D. Weinshall. (2019). “On The Power of Curriculum Learning in Training Deep Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. *Proceedings of Machine Learning Research*. PMLR. 2535–2544. URL: <http://proceedings.mlr.press/v97/hacohen19a.html>.
- Hadsell, R., A. Erkan, P. Sermanet, M. Scoffier, U. Muller, and Y. LeCun. (2008). “Deep belief net learning in a long-range vision system for autonomous off-road driving”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 628–633.
- Hadsell, R., D. Rao, A. A. Rusu, and R. Pascanu. (2020). “Embracing Change: Continual Learning in Deep Neural Networks”. *Trends in Cognitive Sciences*.
- Hamilton, W., Z. Ying, and J. Leskovec. (2017). “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7e9ea9-Paper.pdf>.
- Han, S., J. Pool, S. Narang, H. Mao, E. Gong, S. Tang, E. Elsen, P. Vajda, M. Paluri, J. Tran, *et al.* (2016). “Dsd: Dense-sparse-dense training for deep neural networks”. *arXiv preprint arXiv:1607.04381*.
- Han, S., J. Pool, J. Tran, and W. J. Dally. (2015). “Learning both weights and connections for efficient neural networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*. 1135–1143.
- Happel, B. L. and J. M. Murre. (1994). “Design and evolution of modular neural network architectures”. *Neural networks*. 7(6-7): 985–1004.
- Haugeland, J. (1997). *Mind design II: philosophy, psychology, artificial intelligence*. MIT press.
- Hayes, T. L., K. Kafle, R. Shrestha, M. Acharya, and C. Kanan. (2020). “REMIND Your Neural Network to Prevent Catastrophic Forgetting”. arXiv: [1910.02509 \[cs.LG\]](https://arxiv.org/abs/1910.02509).
- He, K., X. Zhang, S. Ren, and J. Sun. (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.



- Hecht-Nielsen, R. (1992). “Theory of the backpropagation neural network”. In: *Neural networks for perception*. Elsevier. 65–93.
- Hershey, J. R., P. A. Olsen, and S. J. Rennie. (2007). “Variational Kullback-Leibler divergence for hidden Markov models”. In: *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*. IEEE. 323–328.
- Hinton, G., O. Vinyals, and J. Dean. (2015). “Distilling the knowledge in a neural network”. *arXiv preprint arXiv:1503.02531*.
- Hochreiter, S. and J. Schmidhuber. (1997a). “Flat minima”. *Neural computation*. 9(1): 1–42.
- Hochreiter, S. and J. Schmidhuber. (1997b). “Long short-term memory”. *Neural computation*. 9(8): 1735–1780.
- Hou, S., X. Pan, C. C. Loy, Z. Wang, and D. Lin. (2019). “Learning a Unified Classifier Incrementally via Rebalancing”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Houlsby, N., A. Giurghi, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. (2019). “Parameter-efficient transfer learning for NLP”. In: *International Conference on Machine Learning*. PMLR. 2790–2799.
- Hu, H., O. Sener, F. Sha, and V. Koltun. (2020). “Drinking from a Firehose: Continual Learning with Web-scale Natural Language”. arXiv: [2007.09335](https://arxiv.org/abs/2007.09335) [cs.LG].
- Hubara, I., M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. (2016). “Binarized neural networks”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 4114–4122.
- Huszár, F. (2017). “On Quadratic Penalties in Elastic Weight Consolidation”. arXiv: [1712.03847](https://arxiv.org/abs/1712.03847) [stat.ML].
- Ihmels, J., G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. (2002). “Revealing modular organization in the yeast transcriptional network”. *Nature genetics*. 31(4): 370–377.
- Isele, D. and A. Cosgun. (2018). “Selective experience replay for life-long learning”. In: *AAAI Conference on Artificial Intelligence 2018*. Association for the Advancement of Artificial Intelligence (AAAI). 3302–3309.

- Jain, P., B. Kulis, I. S. Dhillon, and K. Grauman. (2008). “Online Metric Learning and Fast Similarity Search.” In: *NIPS*. Vol. 8. Citeseer. 761–768.
- Jastrzebski, S., M. Szymczak, S. Fort, D. Arpit, J. Tabor, K. Cho, and K. Geras. (2020). “The Break-Even Point on Optimization Trajectories of Deep Neural Networks”. In: *International Conference on Learning Representations*.
- Javed, K. and M. White. (2019). “Meta-Learning Representations for Continual Learning”. In: *NeurIPS*.
- Johnson, J., B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. (2017). “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2901–2910.
- Joshi, M., E. Choi, D. S. Weld, and L. Zettlemoyer. (2017). “TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension”. arXiv: [1705.03551](https://arxiv.org/abs/1705.03551) [cs.CL].
- Jung, S., H. Ahn, S. Cha, and T. Moon. (2020). “Continual learning with node-importance based adaptive group sparse regularization”. *arXiv preprint arXiv:2003.13726*.
- Kaplan, J., S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. (2020). “Scaling laws for neural language models”. *arXiv preprint arXiv:2001.08361*.
- Kashtan, N. and U. Alon. (2005). “Spontaneous evolution of modularity and network motifs”. *Proceedings of the National Academy of Sciences*. 102(39): 13773–13778.
- Kashtan, N., E. Noor, and U. Alon. (2007). “Varying environments can speed up evolution”. *Proceedings of the National Academy of Sciences*. 104(34): 13711–13716.
- Kharitonov, E., A. Lee, A. Polyak, Y. Adi, J. Copet, K. Lakhotia, T.-A. Nguyen, M. Riviere, A. Mohamed, E. Dupoux, *et al.* (2021). “Text-free prosody-aware generative spoken language modeling”. *arXiv preprint arXiv:2109.03264*.
- Khetarpal, K., M. Riemer, I. Rish, and D. Precup. (2020). “Towards continual reinforcement learning: A review and perspectives”. *arXiv preprint arXiv:2012.13490*.

- Khot, T., A. Sabharwal, and P. Clark. (2018). “SciTaiL: A Textual Entailment Dataset from Science Question Answering”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Kim, N., S. Feng, C. Gunasekara, and L. Lastras. (2020). “Implicit discourse relation classification: We need to talk about evaluation”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 5404–5414.
- Kipf, T. N. and M. Welling. (2016). “Semi-supervised classification with graph convolutional networks”. *arXiv preprint arXiv:1609.02907*.
- Kirkpatrick, J., R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. (2017). “Overcoming catastrophic forgetting in neural networks”. *Proceedings of the national academy of sciences*. 114(13): 3521–3526.
- Koch, G. R. (2015). “Siamese Neural Networks for One-Shot Image Recognition”. In:
- Koutnik, J., G. Cuccu, J. Schmidhuber, and F. Gomez. (2013). “Evolving large-scale neural networks for vision-based reinforcement learning”. In: *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. 1061–1068.
- Krizhevsky, A., G. Hinton, et al. (2009). “Learning multiple layers of features from tiny images”.
- Krizhevsky, A., I. Sutskever, and G. Hinton. (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems 25*. 1106–1114.
- Krizhevsky, A., I. Sutskever, and G. E. Hinton. (2017). “Imagenet classification with deep convolutional neural networks”. *Communications of the ACM*. 60(6): 84–90.
- Kruszewski, G., I. T. Sorodoc, and T. Mikolov. (2021). “Evaluating Online Continual Learning with {CALM}”. URL: <https://openreview.net/forum?id=vC8hNRk9dOR>.
- Kumar, A., S. Chatterjee, and P. Rai. (2021). “Bayesian structural adaptation for continual learning”. In: *International Conference on Machine Learning*. PMLR. 5850–5860.

- Kumar, A., O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. (2016). “Ask me anything: Dynamic memory networks for natural language processing”. In: *International conference on machine learning*. PMLR. 1378–1387.
- Kurzweil, R., R. Richter, R. Kurzweil, and M. L. Schneider. (1990). *The age of intelligent machines*. Vol. 580. MIT press Cambridge.
- Lakhotia, K., E. Kharitonov, W.-N. Hsu, Y. Adi, A. Polyak, B. Bolte, T.-A. Nguyen, J. Copet, A. Baevski, A. Mohamed, *et al.* (2021). “Generative spoken language modeling from raw audio”. *arXiv preprint arXiv:2102.01192*.
- Laleh, T., M. Faramarzi, I. Rish, and S. Chandar. (2020). “Chaotic Continual Learning”.
- Lample, G., A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou. (2019). “Large memory layers with product keys”. *arXiv preprint arXiv:1907.05242*.
- Lan, Z., M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. (2019). “Albert: A lite bert for self-supervised learning of language representations”. *arXiv preprint arXiv:1909.11942*.
- LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. (1989). “Backpropagation applied to handwritten zip code recognition”. *Neural computation*. 1(4): 541–551.
- LeCun, Y., C. Cortes, and C. Burges. (2010). “MNIST handwritten digit database”. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>. 2.
- Lee, S.-W., J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang. (2017). “Overcoming catastrophic forgetting by incremental moment matching”. *arXiv preprint arXiv:1703.08475*.
- Lee, S. (2017). “Toward Continual Learning for Conversational Agents”. *ArXiv*. abs/1712.09943.
- Leibo, J. Z., C. d. M. d’Autume, D. Zoran, D. Amos, C. Beattie, K. Anderson, A. G. Castañeda, M. Sanchez, S. Green, A. Gruslys, *et al.* (2018). “Psychlab: a psychology laboratory for deep reinforcement learning agents”. *arXiv preprint arXiv:1801.08116*.

- Lesort, T., H. Caselles-Dupre, M. Garcia-Ortiz, A. Stoian, and D. Filliat. (2018). “Generative Models from the perspective of Continual Learning”. arXiv: [1812.09111](https://arxiv.org/abs/1812.09111) [cs.LG].
- Lesort, T., V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez. (2019). “Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges”. arXiv: [1907.00182](https://arxiv.org/abs/1907.00182) [cs.LG].
- Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. (2019). “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”. *arXiv preprint arXiv:1910.13461*.
- Lewis-Kraus, G. (2016). “The Great A.I. Awakening”. *The New York Times*. Dec. URL: <https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html> (accessed on 12/14/2016).
- Li, G., S. C. Hoi, K. Chang, W. Liu, and R. Jain. (2013). “Collaborative online multitask learning”. *IEEE Transactions on Knowledge and Data Engineering*. 26(8): 1866–1876.
- Li, X., Y. Zhou, T. Wu, R. Socher, and C. Xiong. (2019a). “Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting”. arXiv: [1904.00310](https://arxiv.org/abs/1904.00310) [cs.LG].
- Li, Y., L. Zhao, K. Church, and M. Elhoseiny. (2020). “Compositional Language Continual Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=rklnDgHtDS>.
- Li, Y., L. Zhao, J. Wang, and J. Hestness. (2019b). “Compositional generalization for primitive substitutions”. *arXiv preprint arXiv:1910.02612*.
- Li, Y., K. Swersky, and R. Zemel. (2015). “Generative moment matching networks”. In: *International Conference on Machine Learning*. PMLR. 1718–1727.
- Li, Y. and J. Ji. (2021). “Parallel Curriculum Experience Replay in Distributed Reinforcement Learning”. In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 782–789.
- Li, Z. and D. Hoiem. (2017). “Learning without forgetting”. *IEEE transactions on pattern analysis and machine intelligence*. 40(12): 2935–2947.

- Litjens, G., C. I. Sánchez, N. Timofeeva, M. Hermsen, I. Nagtegaal, I. Kovacs, C. Hulsbergen-Van De Kaa, P. Bult, B. Van Ginneken, and J. Van Der Laak. (2016). “Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis”. *Scientific reports*. 6(1): 1–11.
- Liu, B. (2020). “Learning on the Job: Online Lifelong and Continual Learning”. *Proceedings of the AAAI Conference on Artificial Intelligence*. 34(09): 13544–13549. DOI: [10.1609/aaai.v34i09.7079](https://doi.org/10.1609/aaai.v34i09.7079). URL: <https://ojs.aaai.org/index.php/AAAI/article/view/7079>.
- Liu, S., E. Johns, and A. J. Davison. (2019). “End-to-end multi-task learning with attention”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1871–1880.
- Ljung, L. (2010). “Perspectives on system identification”. *Annual Reviews in Control*. 34(1): 1–12. ISSN: 1367-5788. DOI: <https://doi.org/10.1016/j.arcontrol.2009.12.001>. URL: <https://www.sciencedirect.com/science/article/pii/S1367578810000027>.
- Lomonaco, V. and D. Maltoni. (2017). “COrE50: a New Dataset and Benchmark for Continuous Object Recognition”. arXiv: [1705.03550](https://arxiv.org/abs/1705.03550) [cs.CV].
- Lopez-Paz, D. and M. Ranzato. (2017a). “Gradient Episodic Memory for Continual Learning”. arXiv: [1706.08840](https://arxiv.org/abs/1706.08840) [cs.LG].
- Lopez-Paz, D. and M. Ranzato. (2017b). “Gradient episodic memory for continual learning”. In: *Advances in neural information processing systems*. 6467–6476.
- Lüders, B., M. Schläger, and S. Risi. (2016). “Continual learning through evolvable neural turing machines”. In: *Nips 2016 workshop on continual learning and deep networks (cldl 2016)*.
- MacKay, D. J. C. (1992). “A Practical Bayesian Framework for Back-propagation Networks”. *Neural Comput*. 4(3): 448–472. ISSN: 0899-7667. DOI: [10.1162/neco.1992.4.3.448](https://doi.org/10.1162/neco.1992.4.3.448). URL: <https://doi.org/10.1162/neco.1992.4.3.448>.
- Madotto, A., Z. Lin, Z. Zhou, S. Moon, P. Crook, B. Liu, Z. Yu, E. Cho, and Z. Wang. (2020). “Continual Learning in Task-Oriented Dialogue Systems”. arXiv: [2012.15504](https://arxiv.org/abs/2012.15504) [cs.CL].

- Mai, Z., H. Kim, J. Jeong, and S. Sanner. (2020). “Batch-level Experience Replay with Review for Continual Learning”. *arXiv preprint arXiv:2007.05683*.
- Mai, Z., R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. (2021). “Online Continual Learning in Image Classification: An Empirical Survey”. *arXiv e-prints*. Jan., arXiv:2101.10423: arXiv:2101.10423. arXiv: [2101.10423 \[cs.LG\]](#).
- Mallya, A., D. Davis, and S. Lazebnik. (2018). “Piggyback: Adapting a single network to multiple tasks by learning to mask weights”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 67–82.
- Mallya, A. and S. Lazebnik. (2018). “Packnet: Adding multiple tasks to a single network by iterative pruning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7765–7773.
- Malviya, P., B. Ravindran, and S. Chandar. (2021). “TAG: Task-based Accumulated Gradients for Lifelong learning”. arXiv: [2105.05155 \[cs.LG\]](#).
- Mansilla, L., R. Echeveste, D. H. Milone, and E. Ferrante. (2021). “Domain generalization via gradient surgery”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6630–6638.
- Marcus, G. (2018). “Deep learning: A critical appraisal”. *arXiv preprint arXiv:1801.00631*.
- Masana, M., X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer. (2020a). “Class-incremental learning: survey and performance evaluation”. *arXiv preprint arXiv:2010.15277*.
- Masana, M., T. Tuytelaars, and J. van de Weijer. (2020b). “Ternary feature masks: continual learning without any forgetting”. *arXiv preprint arXiv:2001.08714*.
- Masson d’Autume, C. de, S. Ruder, L. Kong, and D. Yogatama. (2019). “Episodic Memory in Lifelong Language Learning”. In: *Advances in Neural Information Processing Systems*. 13143–13152.

- Mathis, A., P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge. (2018). “DeepLabCut: markerless pose estimation of user-defined body parts with deep learning”. *Nature neuroscience*. 21(9): 1281–1289.
- Mathis, M. W. and A. Mathis. (2020). “Deep learning tools for the measurement of animal behavior in neuroscience”. *Current opinion in neurobiology*. 60: 1–11.
- McCann, B., N. S. Keskar, C. Xiong, and R. Socher. (2018). “The natural language decathlon: Multitask learning as question answering”. *arXiv preprint arXiv:1806.08730*.
- McCarthy, J., M. L. Minsky, N. Rochester, and C. E. Shannon. (2006). “A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955”. *AI magazine*. 27(4): 12–12.
- McClelland, J. L., B. L. McNaughton, and R. C. O’Reilly. (1995). “Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory.” *Psychological review*. 102(3): 419.
- McCloskey, M. and N. J. Cohen. (1989). “Catastrophic interference in connectionist networks: The sequential learning problem”. In: *Psychology of learning and motivation*. Vol. 24. Elsevier. 109–165.
- McCulloch, W. S. and W. Pitts. (1943). “A logical calculus of the ideas immanent in nervous activity”. *The bulletin of mathematical biophysics*. 5(4): 115–133.
- McNamara, C. G., Á. Tejero-Cantero, S. Trouche, N. Campo-Urriza, and D. Dupret. (2014). “Dopaminergic neurons promote hippocampal reactivation and spatial memory persistence”. *Nature neuroscience*. 17(12): 1658–1660.
- Mehta, S. V., D. Patil, S. Chandar, and E. Strubell. (2021). “An empirical investigation of the role of pre-training in lifelong learning”. *arXiv preprint arXiv:2112.09153*.
- Mermillod, M., A. Bugajska, and P. Bonin. (2013). “The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects”. *Frontiers in psychology*. 4: 504.



- Meunier, D., R. Lambiotte, and E. T. Bullmore. (2010). “Modular and hierarchically modular organization of brain networks”. *Frontiers in neuroscience*. 4: 200.
- Mi, F., L. Chen, M. Zhao, M. Huang, and B. Faltings. (2020). “Continual Learning for Natural Language Generation in Task-oriented Dialog Systems”. arXiv: [2010.00910](https://arxiv.org/abs/2010.00910) [cs.CL].
- Miller, A., A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. (2016). “Key-value memory networks for directly reading documents”. *arXiv preprint arXiv:1606.03126*.
- Minsky, M. and S. A. Papert. (1972). “Artificial intelligence progress report”.
- Mirhoseini, A., A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, *et al.* (2021). “A graph placement methodology for fast chip design”. *Nature*. 594(7862): 207–212.
- Mirzadeh, S. I., M. Farajtabar, and H. Ghasemzadeh. (2020a). “Dropout as an implicit gating mechanism for continual learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 232–233.
- Mirzadeh, S. I., M. Farajtabar, R. Pascanu, and H. Ghasemzadeh. (2020b). “Understanding the Role of Training Regimes in Continual Learning”. arXiv: [2006.06958](https://arxiv.org/abs/2006.06958) [cs.LG].
- Mishra, N., M. Rohaninejad, X. Chen, and P. Abbeel. (2018). “A Simple Neural Attentive Meta-Learner”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=B1DmUzWAW>.
- Mitchell, T., W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, *et al.* (2018). “Never-ending learning”. *Communications of the ACM*. 61(5): 103–115.
- Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Belle-mare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. (2015). “Human-level control through deep reinforcement learning”. *Nature*. 518(7540): 529–533. ISSN: 00280836. URL: <http://dx.doi.org/10.1038/nature14236>.

- Mostafazadeh, N., N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. (2016). “A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics. 839–849. DOI: [10.18653/v1/N16-1098](https://doi.org/10.18653/v1/N16-1098). URL: <https://www.aclweb.org/anthology/N16-1098>.
- Munkhdalai, T. and H. Yu. (2017). “Meta Networks”. *Proceedings of machine learning research*. 70: 2554–2563.
- Murugesan, K. and J. Carbonell. (2017). “Self-paced multitask learning with shared knowledge”. *arXiv preprint arXiv:1703.00977*.
- Nagabandi, A., I. Clavera, S. Liu, R. Fearing, P. Abbeel, S. Levine, and C. Finn. (2019). “Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning”. *arXiv: Learning*.
- Narvekar, S. (2017). “Curriculum Learning in Reinforcement Learning.” In: *IJCAI*. 5195–5196.
- Narvekar, S., B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. (2020). “Curriculum learning for reinforcement learning domains: A framework and survey”. *arXiv preprint arXiv:2003.04960*.
- Narvekar, S. and P. Stone. (2018). “Learning curriculum policies for reinforcement learning”. *arXiv preprint arXiv:1812.00285*.
- Naumov, M., D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini, *et al.* (2019). “Deep learning recommendation model for personalization and recommendation systems”. *arXiv preprint arXiv:1906.00091*.
- Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. (2011). “Reading digits in natural images with unsupervised feature learning”.
- Newell, A. and J. Shaw. (1959). “A variety of intelligent learning in a general problem solver”. *RAND Report P-1742, dated July*. 6.
- Newman, M. E. (2006). “Modularity and community structure in networks”. *Proceedings of the national academy of sciences*. 103(23): 8577–8582.
- Nguyen, C. V., Y. Li, T. D. Bui, and R. E. Turner. (2017). “Variational Continual Learning”. arXiv: [1710.10628](https://arxiv.org/abs/1710.10628) [stat.ML].

- Nichol, A., J. Achiam, and J. Schulman. (2018). “On first-order meta-learning algorithms”. *arXiv preprint arXiv:1803.02999*.
- O’Reilly, R. C. and K. A. Norman. (2002). “Hippocampal and neocortical contributions to memory: Advances in the complementary learning systems framework”. *Trends in cognitive sciences*. 6(12): 505–510.
- Ólafsdóttir, H. F., C. Barry, A. B. Saleem, D. Hassabis, and H. J. Spiers. (2015). “Hippocampal place cells construct reward related sequences through unexplored space”. *Elife*. 4: e06063.
- Özgün, S., A.-M. Rickmann, A. G. Roy, and C. Wachinger. (2020). “Importance driven continual learning for segmentation across domains”. In: *International Workshop on Machine Learning in Medical Imaging*. Springer. 423–433.
- Pan, S. J. and Q. Yang. (2009). “A survey on transfer learning”. *IEEE Transactions on knowledge and data engineering*. 22(10): 1345–1359.
- Parascandolo, G., N. Kilbertus, M. Rojas-Carulla, and B. Schölkopf. (2018). “Learning Independent Causal Mechanisms”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. *Proceedings of Machine Learning Research*. Stockholmsmässan, Stockholm Sweden: PMLR. 4036–4044. URL: <http://proceedings.mlr.press/v80/parascandolo18a.html>.
- Parisi, G. I. and V. Lomonaco. (2020). “Online Continual Learning on Sequences”. *arXiv e-prints*. Mar., arXiv:2003.09114: arXiv:2003.09114. arXiv: [2003.09114 \[cs.LG\]](https://arxiv.org/abs/2003.09114).
- Pentina, A., V. Sharmanska, and C. H. Lampert. (2015). “Curriculum learning of multiple tasks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5492–5500.
- Peterson, G. B. (2004). “A day of great illumination: BF Skinner’s discovery of shaping”. *Journal of the experimental analysis of behavior*. 82(3): 317–328.
- Pfau, D., J. S. Spencer, A. G. D. G. Matthews, and W. M. C. Foulkes. (2020). “Ab initio solution of the many-electron Schrödinger equation with deep neural networks”. *Phys. Rev. Research*. 2(3): 033429. DOI: [10.1103/PhysRevResearch.2.033429](https://doi.org/10.1103/PhysRevResearch.2.033429). URL: <https://link.aps.org/doi/10.1103/PhysRevResearch.2.033429>.

- Pfeiffer, J., A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. (2021). “AdapterFusion: Non-destructive task composition for transfer learning”. *EACL*.
- Pfeiffer, J., I. Vulić, I. Gurevych, and S. Ruder. (2020). “MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 7654–7673.
- Pham, Q., C. Liu, D. Sahoo, and S. HOI. (2021). “Contextual Transformation Networks for Online Continual Learning”. In: *International Conference on Learning Representations*. URL: [https://openreview.net/forum?id=zx\\_uX-BO7CH](https://openreview.net/forum?id=zx_uX-BO7CH).
- Pinker, S. (1994). “The Language Instinct. How the Mind Creates Language”.
- Pinker, S. (2005). “So how does the mind work?” *Mind & Language*. 20(1): 1–24.
- Pless, R. and R. Souvenir. (2009). “A survey of manifold learning for images”. *IPSN Transactions on Computer Vision and Applications*. 1: 83–94.
- Poliak, A., A. Haldar, R. Rudinger, J. E. Hu, E. Pavlick, A. S. White, and B. Van Durme. (2018). “Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 67–81.
- Polyak, A., Y. Adi, J. Copet, E. Kharitonov, K. Lakhotia, W.-N. Hsu, A. Mohamed, and E. Dupoux. (2021). “Speech Resynthesis from Discrete Disentangled Self-Supervised Representations”. *ArXiv*. abs/2104.00355.
- Portelas, R., C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. (2020). “Automatic curriculum learning for deep rl: A short survey”. *arXiv preprint arXiv:2003.04664*.
- Prasad, R., B. Webber, A. Lee, and A. Joshi. (2019). “Penn Discourse Treebank Version 3.0”. In: *LDC2019T05*. Philadelphia: Linguistic Data Consortium.

- Purushwalkam, S., M. Nickel, A. Gupta, and M. Ranzato. (2019). “Task-driven modular networks for zero-shot compositional learning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3593–3602.
- Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. (2019). “Language models are unsupervised multitask learners”. *OpenAI Blog*. 1(8): 9.
- Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. (2019). “Exploring the limits of transfer learning with a unified text-to-text transformer”. *arXiv preprint arXiv:1910.10683*.
- Rajasegaran, J., M. Hayat, S. Khan, F. S. Khan, L. Shao, and M.-H. Yang. (2019a). “An adaptive random path selection approach for incremental learning”. *arXiv preprint arXiv:1906.01120*.
- Rajasegaran, J., M. Hayat, S. H. Khan, F. S. Khan, and L. Shao. (2019b). “Random Path Selection for Continual Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2019/file/83da7c539e1ab4e759623c38d8737e9e-Paper.pdf>.
- Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang. (2016). “Squad: 100,000+ questions for machine comprehension of text”. *arXiv preprint arXiv:1606.05250*.
- Ramalho, T. and M. Garnelo. (2019). “Adaptive Posterior Learning: few-shot learning with a surprise-based memory module”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=ByeSdsC9Km>.
- Ramirez, S., X. Liu, P.-A. Lin, J. Suh, M. Pignatelli, R. L. Redondo, T. J. Ryan, and S. Tonegawa. (2013). “Creating a false memory in the hippocampus”. *Science*. 341(6144): 387–391.
- Ramos, J. *et al.* (2003). “Using tf-idf to determine word relevance in document queries”. In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. No. 1. Citeseer. 29–48.
- Rannen, A., R. Aljundi, M. B. Blaschko, and T. Tuytelaars. (2017). “Encoder Based Lifelong Learning”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

- Rao, D., F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell. (2019). “Continual unsupervised representation learning”. In: *Advances in Neural Information Processing Systems*. 7647–7657.
- Rastogi, A., X. Zang, S. Sunkara, R. Gupta, and P. Khaitan. (2020). “Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset”. arXiv: [1909.05855](https://arxiv.org/abs/1909.05855) [cs.CL].
- Ratcliff, R. (1990). “Connectionist models of recognition memory: constraints imposed by learning and forgetting functions.” *Psychological review*. 97(2): 285.
- Ravi, S. and H. Larochelle. (2017). “Optimization as a Model for Few-Shot Learning”. In: *ICLR*.
- Rebuffi, S.-A., A. Kolesnikov, G. Sperl, and C. H. Lampert. (2017). “iCaRL: Incremental Classifier and Representation Learning”. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July. DOI: [10.1109/cvpr.2017.587](https://doi.org/10.1109/cvpr.2017.587). URL: <http://dx.doi.org/10.1109/CVPR.2017.587>.
- Ren, S., K. He, R. Girshick, and J. Sun. (2015). “Faster r-cnn: Towards real-time object detection with region proposal networks”. *Advances in neural information processing systems*. 28: 91–99.
- Rich, E. and K. Knight. (1992). *Artificial Intelligence: Instructor’s Manual*. McGraw-Hill.
- Riemer, M., I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro. (2019). “Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference”. In: *International Conference on Learning Representations*.
- Ritter, H., A. Botev, and D. Barber. (2018a). “Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2018/file/f31b20466ae89669f9741e047487eb37-Paper.pdf>.
- Ritter, S., J. X. Wang, Z. Kurth-Nelson, S. M. Jayakumar, C. Blundell, R. Pascanu, and M. Botvinick. (2018b). “Been There, Done That: Meta-Learning with Episodic Recall”. In: *ICML*.

- Roady, R., T. L. Hayes, H. Vaidya, and C. Kanan. (2020). “Stream-51: Streaming Classification and Novelty Detection From Videos”. In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Rosenbaum, C., T. Klinger, and M. Riemer. (2017). “Routing networks: Adaptive selection of non-linear functions for multi-task learning”. *arXiv preprint arXiv:1711.01239*.
- Ruder, S. (2017). “An overview of multi-task learning in deep neural networks”. *arXiv preprint arXiv:1706.05098*.
- Russell, S. and P. Norvig. (2005). “AI a modern approach”. *Learning*. 2(3): 4.
- Rusu, A. A., N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. (2016). “Progressive neural networks”. *arXiv preprint arXiv:1606.04671*.
- Saha, G., I. Garg, and K. Roy. (2021). “Gradient projection memory for continual learning”. *arXiv preprint arXiv:2103.09762*.
- Santoro, A., S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. (2016). “Meta-Learning with Memory-Augmented Neural Networks”. In: *ICML*.
- Santoro, A., D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. (2017). “A simple neural network module for relational reasoning”. In: *Advances in neural information processing systems*. 4967–4976.
- Sarafianos, N., T. Giannakopoulos, C. Nikou, and I. A. Kakadiaris. (2017). “Curriculum learning for multi-task classification of visual attributes”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2608–2615.
- Saravia, E., H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen. (2018). “CAREER: Contextualized Affect Representations for Emotion Recognition”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics. 3687–3697. DOI: [10.18653/v1/D18-1404](https://doi.org/10.18653/v1/D18-1404). URL: <https://www.aclweb.org/anthology/D18-1404>.

- Satariano, A. and C. Metz. (2020). “A Warehouse Robot Learns to Sort Out the Tricky Stuff”. *The New York Times*. Jan. URL: <https://www.nytimes.com/2020/01/29/technology/warehouse-robot.html> (accessed on 01/29/2020).
- Schalkoff, R. (1991). “Artificial Intelligence: An Engineering Approach”.
- Schaul, T., H. van Hasselt, J. Modayil, M. White, A. White, P.-L. Bacon, J. Harb, S. Mourad, M. Bellemare, and D. Precup. (2018). “The barbados 2018 list of open issues in continual learning”. *arXiv preprint arXiv:1811.07004*.
- Schmidhuber, J. (1987). “Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook”. *PhD thesis*. Technische Universität München.
- Schmidt, J., M. R. Marques, S. Botti, and M. A. Marques. (2019). “Recent advances and applications of machine learning in solid-state materials science”. *npj Computational Materials*. 5(1): 1–36.
- Schneider, S., A. Baevski, R. Collobert, and M. Auli. (2019). “wav2vec: Unsupervised pre-training for speech recognition”. *arXiv preprint arXiv:1904.05862*.
- Schrittwieser, J., I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.* (2020). “Mastering atari, go, chess and shogi by planning with a learned model”. *Nature*. 588(7839): 604–609.
- Schwarz, J., W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. (2018). “Progress and compress: A scalable framework for continual learning”. In: *International Conference on Machine Learning*. PMLR. 4528–4537.
- Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. (2017). “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 618–626.
- Senior, A. W., R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žíředek, A. W. Nelson, A. Bridgland, *et al.* (2020). “Improved protein structure prediction using potentials from deep learning”. *Nature*. 577(7792): 706–710.



- Serra, J., D. Suris, M. Miron, and A. Karatzoglou. (2018). “Overcoming catastrophic forgetting with hard attention to the task”. In: *International Conference on Machine Learning*. PMLR. 4548–4557.
- Shalev-Shwartz, S. (2012). “Online Learning and Online Convex Optimization”. *Foundations and Trends® in Machine Learning*. 4(2): 107–194. ISSN: 1935-8237. DOI: [10.1561/22000000018](https://doi.org/10.1561/22000000018). URL: <http://dx.doi.org/10.1561/22000000018>.
- Shalev-shwartz, S. and P. Y. Singer. (2007). “Online learning: Theory, algorithms, and applications”. *Tech. rep.*
- Shalev-Shwartz, S., Y. Singer, and A. Y. Ng. (2004). “Online and batch learning of pseudo-metrics”. In: *Proceedings of the twenty-first international conference on Machine learning*. 94.
- SHARKEY, A. J. C. (1996). “On combining artificial neural nets”. *Connection science*. 8(3-4): 299–314.
- SHARKEY, A. J. C. (1997). “Modularity, combining and artificial neural nets”. *Connection Science*. 9(1): 3–10.
- Shashkevich, A. (2019). “Stanford researcher examines earliest concepts of artificial intelligence, robots in ancient myths”. *Stanford News*. Feb. URL: <https://news.stanford.edu/2019/02/28/ancient-myths-reveal-early-fantasies-artificial-life/> (accessed on 02/28/2019).
- She, Q., F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang, F. Qiao, and R. H. M. Chan. (2020). “OpenLORIS-Object: A Robotic Vision Dataset and Benchmark for Lifelong Deep Learning”. In: *2020 International Conference on Robotics and Automation (ICRA)*. 4767–4773.
- Al-Shedivat, M., T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. (2018). “Continuous Adaptation via Meta-Learning in Non-stationary and Competitive Environments”. *ArXiv*. abs/1710.03641.
- Shin, H., J. K. Lee, J. Kim, and J. Kim. (2017). “Continual learning with deep generative replay”. In: *Advances in Neural Information Processing Systems*. 2990–2999.
- Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.* (2016). “Mastering the game of Go with deep neural networks and tree search”. *nature*. 529(7587): 484.

- Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.* (2018). “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. *Science*. 362(6419): 1140–1144.
- Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.* (2017). “Mastering the game of go without human knowledge”. *nature*. 550(7676): 354–359.
- Skinner, B. F. (1958). “Reinforcement today.” *American Psychologist*. 13(3): 94.
- Snell, J., K. Swersky, and R. Zemel. (2017). “Prototypical Networks for Few-shot Learning”. In: *NIPS*.
- Socher, R., A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. (2013). “Recursive deep models for semantic compositionality over a sentiment treebank”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.
- Sodhani, S., S. Chandar, and Y. Bengio. (2018). “On Training Recurrent Neural Networks for Lifelong Learning”. *arXiv preprint arXiv:1811.07017*.
- Sodhani, S., S. Chandar, and Y. Bengio. (2020). “Toward training recurrent neural networks for lifelong learning”. *Neural computation*. 32(1): 1–35.
- Sodhani, S., M. Qu, and J. Tang. (2019). “Attending Over Triads for Learning Signed Network Embedding”. *Frontiers in big Data*. 2: 6.
- Sodhani, S., A. Zhang, and J. Pineau. (2021). “Multi-Task Reinforcement Learning with Context-based Representations”. *arXiv preprint arXiv:2102.06177*.
- Solomonoff, R. J. (1989). “A system for incremental learning based on algorithmic probability”. In: *Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition*. 515–527.
- Sorower, M. S. (2010). “A literature survey on algorithms for multi-label learning”. *Tech. rep.*
- Spelke, E. S. (1990). “Principles of object perception”. *Cognitive science*. 14(1): 29–56.

- Spelke, E. S. and K. D. Kinzler. (2007). “Core knowledge”. *Developmental science*. 10(1): 89–96.
- Sprechmann, P., S. Jayakumar, J. Rae, A. Pritzel, A. P. Badia, B. Uria, O. Vinyals, D. Hassabis, R. Pascanu, and C. Blundell. (2018). “Memory-based Parameter Adaptation”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=rkfOvGbcCW>.
- Stab, C., T. Miller, B. Schiller, P. Rai, and I. Gurevych. (2018). “Cross-topic Argument Mining from Heterogeneous Sources”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3664–3674.
- Standley, T., A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. (2020). “Which tasks should be learned together in multi-task learning?” In: *International Conference on Machine Learning*. PMLR. 9120–9132.
- Stickgold, R. and M. P. Walker. (2007). “Sleep-dependent memory consolidation and reconsolidation”. *Sleep medicine*. 8(4): 331–343.
- Stickland, A. C. and I. Murray. (2019). “Bert and pals: Projected attention layers for efficient adaptation in multi-task learning”. In: *International Conference on Machine Learning*. PMLR. 5986–5995.
- Stojanov, S., S. Mishra, N. A. Thai, N. Dhandra, A. Humayun, C. Yu, L. B. Smith, and J. M. Rehg. (2019). “Incremental Object Learning From Contiguous Views”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Sukhbaatar, S., J. Weston, R. Fergus, *et al.* (2015). “End-to-end memory networks”. In: *Advances in neural information processing systems*. 2440–2448.
- Sun, F.-K., C.-H. Ho, and H.-Y. Lee. (2020). “LAMOL: LAnguage MOdeling for Lifelong Language Learning”. In: *International Conference on Learning Representations*.
- Sung, F., Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales. (2018). “Learning to Compare: Relation Network for Few-Shot Learning”. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*: 1199–1208.

- Sussmann, H. J. (1992). “Uniqueness of the weights for minimal feedforward nets with a given input-output map”. *Neural networks*. 5(4): 589–593.
- Suteu, M. and Y. Guo. (2019). “Regularizing Deep Multi-Task Networks using Orthogonal Gradients”. *arXiv preprint arXiv:1912.06844*.
- Swaroop, S., C. V. Nguyen, T. D. Bui, and R. E. Turner. (2019). “Improving and Understanding Variational Continual Learning”. arXiv: [1905.02099](https://arxiv.org/abs/1905.02099) [stat.ML].
- Swevers, J., C. Ganseman, D. B. Tukel, J. De Schutter, and H. Van Brussel. (1997). “Optimal robot excitation and identification”. *IEEE transactions on robotics and automation*. 13(5): 730–740.
- Tan, C., F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. (2018). “A survey on deep transfer learning”. In: *International conference on artificial neural networks*. Springer. 270–279.
- Tang, B. and D. S. Matteson. (2021). “Graph-Based Continual Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=HHSEKOnPvaO>.
- Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. (2015). “LINE: Large-Scale Information Network Embedding”. In: *Proceedings of the 24th International Conference on World Wide Web. WWW '15*. Florence, Italy: International World Wide Web Conferences Steering Committee. 1067–1077. ISBN: 9781450334693. DOI: [10.1145/2736277.2741093](https://doi.org/10.1145/2736277.2741093). URL: <https://doi.org/10.1145/2736277.2741093>.
- Thrun, S. (1996). “Explanation-based neural network learning”. In: *Explanation-Based Neural Network Learning*. Springer. 19–48.
- Thrun, S. (2012). *Explanation-based neural network learning: A lifelong learning approach*. Vol. 357. Springer Science & Business Media.
- Thrun, S. and L. Pratt. (1998). “Learning to Learn: Introduction and Overview”. In: *Learning to Learn*. Ed. by S. Thrun and L. Pratt. Boston, MA: Springer US. 3–17. ISBN: 978-1-4615-5529-2. DOI: [10.1007/978-1-4615-5529-2\\_1](https://doi.org/10.1007/978-1-4615-5529-2_1). URL: [https://doi.org/10.1007/978-1-4615-5529-2\\_1](https://doi.org/10.1007/978-1-4615-5529-2_1).

- Titsias, M. K., J. Schwarz, A. G. de G. Matthews, R. Pascanu, and Y. W. Teh. (2020). “Functional Regularisation for Continual Learning with Gaussian Processes”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=HkxCzeHFDB>.
- Tokdar, S. T. and R. E. Kass. (2010). “Importance sampling: a review”. *Wiley Interdisciplinary Reviews: Computational Statistics*. 2(1): 54–60.
- Tomašev, N., X. Glorot, J. W. Rae, M. Zielinski, H. Askham, A. Saraiva, A. Mottram, C. Meyer, S. Ravuri, I. Protsyuk, *et al.* (2019). “A clinically applicable approach to continuous prediction of future acute kidney injury”. *Nature*. 572(7767): 116–119.
- Toneva, M., A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon. (2019). “An Empirical Study of Example Forgetting during Deep Neural Network Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=BJlxm30cKm>.
- Torrey, L. and J. Shavlik. (2010). “Transfer learning”. In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global. 242–264.
- Tulving, E. (1985). “How many memory systems are there?” *American psychologist*. 40(4): 385.
- TURING, A. M. (1950). “I.—COMPUTING MACHINERY AND INTELLIGENCE”. *Mind*. LIX(236): 433–460. ISSN: 0026-4423. DOI: [10.1093/mind/LIX.236.433](https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf). eprint: <https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf>. URL: <https://doi.org/10.1093/mind/LIX.236.433>.
- Van Overschee, P. and B. De Moor. (2012). *Subspace identification for linear systems: Theory—Implementation—Applications*. Springer Science & Business Media.
- Vapnik, V. (1991). “Principles of Risk Minimization for Learning Theory”. In: *Proceedings of the 4th International Conference on Neural Information Processing Systems. NIPS’91*. Denver, Colorado: Morgan Kaufmann Publishers Inc. 831–838. ISBN: 1558602224.

- Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. (2017). “Attention is all you need”. In: *Advances in neural information processing systems*. 5998–6008.
- Ven, G. M. van de, H. T. Siegelmann, and A. S. Tolias. (2020). “Brain-inspired replay for continual learning with artificial neural networks”. *Nature communications*. 11(1): 1–14.
- Ven, G. M. van de and A. S. Tolias. (2019a). “Generative replay with feedback connections as a general strategy for continual learning”. arXiv: [1809.10635](https://arxiv.org/abs/1809.10635) [cs.LG].
- Ven, G. M. van de and A. S. Tolias. (2019b). “Three scenarios for continual learning”. arXiv: [1904.07734](https://arxiv.org/abs/1904.07734) [cs.LG].
- Veniat, T., L. Denoyer, and M. Ranzato. (2021). “Efficient Continual Learning with Modular Networks and Task-Driven Priors”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=EKV158tSfww>.
- Vezhnevets, A. S., S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. (2017). “Feudal networks for hierarchical reinforcement learning”. *arXiv preprint arXiv:1703.01161*.
- Vincent, J. (2021). “Alphabet is putting its prototype robots to work cleaning up around Google’s offices”. *The Verge*. Nov. URL: <https://www.theverge.com/2021/11/19/22791267/alphabet-google-everyday-robot-project-cleaning-office-prototype> (accessed on 11/19/2021).
- Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.* (2019). “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. *Nature*. 575(7782): 350–354.
- Vinyals, O., C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. (2016). “Matching Networks for One Shot Learning”. In: *NIPS*.
- Vitter, J. S. (1985). “Random sampling with a reservoir”. *ACM Transactions on Mathematical Software (TOMS)*. 11(1): 37–57.
- Wagner, G. P., M. Pavlicev, and J. M. Cheverud. (2007). “The road to modularity”. *Nature Reviews Genetics*. 8(12): 921–931.
- Wang, A., A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. (2018). “GLUE: A multi-task benchmark and analysis platform for natural language understanding”. *arXiv preprint arXiv:1804.07461*.

- Wang, J., C. Lan, C. Liu, Y. Ouyang, W. Zeng, and T. Qin. (2021). “Generalizing to Unseen Domains: A Survey on Domain Generalization”. *arXiv preprint arXiv:2103.03097*.
- Wang, J., Y. Sun, W. Zhang, I. Thomas, S. Duan, and Y. Shi. (2016). “Large-scale online multitask learning and decision making for flexible manufacturing”. *IEEE Transactions on Industrial Informatics*. 12(6): 2139–2147.
- Wang, X., Y. Chen, and W. Zhu. (2020a). “A Survey on Curriculum Learning”. *arXiv preprint arXiv:2010.13166*.
- Wang, Z., S. V. Mehta, B. Póczos, and J. G. Carbonell. (2020b). “Efficient Meta Lifelong-Learning with Limited Memory”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 535–548.
- Warstadt, A., A. Singh, and S. R. Bowman. (2019). “Neural network acceptability judgments”. *Transactions of the Association for Computational Linguistics*. 7: 625–641.
- Weinshall, D., G. Cohen, and D. Amir. (2018). “Curriculum learning by transfer learning: Theory and experiments with deep networks”. In: *International Conference on Machine Learning*. PMLR. 5238–5246.
- Weiss, K., T. M. Khoshgoftaar, and D. Wang. (2016). “A survey of transfer learning”. *Journal of Big data*. 3(1): 1–40.
- Welling, M. (2009). “Herding dynamical weights to learn”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. 1121–1128.
- Wellman, H. M. and S. A. Gelman. (1992). “Cognitive development: Foundational theories of core domains”. *Annual review of psychology*. 43(1): 337–375.
- Wen, T.-H., D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young. (2017). “A Network-based End-to-End Trainable Task-oriented Dialogue System”. arXiv: [1604.04562](https://arxiv.org/abs/1604.04562) [cs.CL].
- Weston, J., A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. (2015). “Towards ai-complete question answering: A set of prerequisite toy tasks”. *arXiv preprint arXiv:1502.05698*.

- Williams, A., N. Nangia, and S. Bowman. (2018). “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 1112–1122.
- Winston, P. H. (1992). “Artificial intelligence”.
- Woo, E. (2014). “John McCarthy dies at 84; the father of artificial intelligence”. *The LA Times*. Mar. URL: <https://www.latimes.com/local/obituaries/la-me-john-mccarthy-20111027-story.html> (accessed on 03/20/2014).
- Wu, C.-S., A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung. (2019a). “Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. 808–819. DOI: [10.18653/v1/P19-1078](https://doi.org/10.18653/v1/P19-1078). URL: <https://aclanthology.org/P19-1078>.
- Wu, P., S. C. H. Hoi, P. Zhao, C. Miao, and Z.-Y. Liu. (2016). “Online Multi-Modal Distance Metric Learning with Application to Image Retrieval”. *IEEE Transactions on Knowledge and Data Engineering*. 28(2): 454–467. DOI: [10.1109/TKDE.2015.2477296](https://doi.org/10.1109/TKDE.2015.2477296).
- Wu, Y., Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. (2019b). “Large Scale Incremental Learning”. arXiv: [1905.13260 \[cs.CV\]](https://arxiv.org/abs/1905.13260).
- Xiao, H., K. Rasul, and R. Vollgraf. (2017). “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. *arXiv preprint arXiv:1708.07747*.
- Xie, S., R. Girshick, P. Dollár, Z. Tu, and K. He. (2017). “Aggregated residual transformations for deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1492–1500.
- Xie, Z., I. Sato, and M. Sugiyama. (2021). “A Diffusion Theory For Deep Learning Dynamics: Stochastic Gradient Descent Exponentially Favors Flat Minima”. In: *International Conference on Learning Representations*.



- Xiong, F., B. Sun, X. Yang, H. Qiao, K. Huang, A. Hussain, and Z. Liu. (2018). “Guided policy search for sequential multitask learning”. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 49(1): 216–226.
- Xu, F., K. Dewar, and A. Perfors. (2009). “Induction, overhypotheses, and the shape bias: Some arguments and evidence for rational constructivism”. *The origins of object knowledge*: 263–284.
- Yim, J., R. Chopra, T. Spitz, J. Winkens, A. Obika, C. Kelly, H. Askham, M. Lukic, J. Huemer, K. Fasler, *et al.* (2020). “Predicting conversion to wet age-related macular degeneration using deep learning”. *Nature Medicine*. 26(6): 892–899.
- Ying, W., Y. Zhang, J. Huang, and Q. Yang. (2018). “Transfer learning via learning to transfer”. In: *International conference on machine learning*. PMLR. 5085–5094.
- Yoon, J., E. Yang, J. Lee, and S. J. Hwang. (2018). “Lifelong Learning with Dynamically Expandable Networks”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=Sk7KsfW0->.
- Yu, L., Z. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg. (2018). “Mattnet: Modular attention network for referring expression comprehension”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1307–1315.
- Yu, T., S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. (2020). “Gradient surgery for multi-task learning”. *arXiv preprint arXiv:2001.06782*.
- Yu, W., J. Tan, C. K. Liu, and G. Turk. (2017). “Preparing for the Unknown: Learning a Universal Policy with Online System Identification”. In: *Robotics: Science and Systems*.
- Zadeh, L. (1956). “On the identification problem”. *IRE Transactions on Circuit Theory*. 3(4): 277–281.
- Zamir, A. R., A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. (2018). “Taskonomy: Disentangling task transfer learning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3712–3722.

- Zenke, F., B. Poole, and S. Ganguli. (2017). “Continual Learning Through Synaptic Intelligence”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. *Proceedings of Machine Learning Research*. International Convention Centre, Sydney, Australia: PMLR. 3987–3995. URL: <http://proceedings.mlr.press/v70/zenke17a.html>.
- Zhai, A., D. Kislyuk, Y. Jing, M. Feng, E. Tzeng, J. Donahue, Y. L. Du, and T. Darrell. (2017). “Visual discovery at pinterest”. In: *Proceedings of the 26th International Conference on World Wide Web Companion*. 515–524.
- Zhang, A., S. Sodhani, K. Khetarpal, and J. Pineau. (2020a). “Learning Robust State Abstractions for Hidden-Parameter Block MDPs”. *arXiv preprint arXiv:2007.07206*.
- Zhang, A., S. Sodhani, K. Khetarpal, and J. Pineau. (2020b). “Multi-Task Reinforcement Learning as a Hidden-Parameter Block MDP”. *arXiv preprint arXiv:2007.07206*.
- Zhang, A., Y. Wu, and J. Pineau. (2018). “Natural environment benchmarks for reinforcement learning”. *arXiv preprint arXiv:1811.06032*.
- Zhang, H., C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, *et al.* (2020c). “Resnest: Split-attention networks”. *arXiv preprint arXiv:2004.08955*.
- Zhang, J., J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo. “Class-incremental learning via deep model consolidation”.
- Zhang, X., J. Zhao, and Y. LeCun. (2015). “Character-level convolutional networks for text classification”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*. 649–657.
- Zhang, Y. and Q. Yang. (2017). “A survey on multi-task learning”. *arXiv preprint arXiv:1707.08114*.
- Zhang, Z., P. Luo, C. C. Loy, and X. Tang. (2014). “Facial landmark detection by deep multi-task learning”. In: *European conference on computer vision*. Springer. 94–108.
- Zhang, Z., J. Yang, and H. Zhao. (2020d). “Retrospective reader for machine reading comprehension”. *arXiv preprint arXiv:2001.09694*.

- Zhao, P., S. C. H. Hoi, and R. Jin. (2011). “Double Updating Online Learning”. *J. Mach. Learn. Res.* 12(null): 1587–1615. ISSN: 1532-4435.
- Zhao, P., S. C. Hoi, J. Wang, and B. Li. (2014). “Online Transfer Learning”. *Artificial Intelligence.* 216: 76–102. ISSN: 0004-3702. DOI: <https://doi.org/10.1016/j.artint.2014.06.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0004370214000800>.
- Zhong, C., Z. Cui, R. Wang, and W.-S. Zheng. (2021). “Discriminative Distillation to Reduce Class Confusion in Continual Learning”. *arXiv preprint arXiv:2108.05187*.
- Zhu, S., A. Kimmell, K. E. Bekris, and A. Boularias. (2017). “Fast model identification via physics engines for data-efficient policy search”. *arXiv preprint arXiv:1710.08893*.
- Zhuang, F., Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. (2020). “A comprehensive survey on transfer learning”. *Proceedings of the IEEE.* 109(1): 43–76.
- Zinkevich, M. (2003). “Online Convex Programming and Generalized Infinitesimal Gradient Ascent”. In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning. ICML’03*. Washington, DC, USA: AAAI Press. 928–935. ISBN: 1577351894.